Académie universitaire Wallonie–Europe
Université de Liège — Faculté des Sciences Appliquées
Collège de doctorat en Électricité, électronique et informatique

Indirect Quadrangular Mesh Generation and Validation of Curved Finite Elements

Doctoral Dissertation presented by

Amaury Johnen

in fulfilment of the requirements for the degree of
Docteur en Sciences de l'Ingénieur

Frebruary 2016

Thesis committee:

Prof. Éric Bechet (Université de Liège)
Prof. Damien Ernst (Université de Liège), President
Prof. Pascal Frey (Université de Paris 6)
Prof. Christophe Geuzaine (Université de Liège), Advisor
Dr. Bruno Lévy (INRIA Nancy)
Prof. Jean-François Remacle (Université catholique de Louvain)
Prof. Josep Sarrate (Université Polytechnique de Catalogne)

Abstract

Among the different types of 3D finite element meshes, hexahedral meshes present properties that can be highly desirable, such as alignment with physical features or a lower computational cost. For this reason and despite the maturity of the tetrahedral mesh generators, hexahedral mesh generation has always been a prolific research domain. Yet, there exists currently no robust algorithm capable of generating conformal all-hexahedral meshes with prescribed input size field on any arbitrary geometry. One difficulty that remains is that there exists no method to robustly assert that a hexahedron is valid. Indeed, linear hexahedra can be folded (tangled) in the same way than curvilinear tetrahedra.

This thesis addresses two subjects. First, two original quadrangular mesh generation techniques are investigated, with the aim to generalize them to 3D. Both are indirect methods and thus consider the problem of combining pairs of triangles of an initial input triangular mesh. The first technique, called *Blossom-Quad*, computes the optimal solution of this problem with respect to a given quality criterion. As for any indirect method, the quality of the solution strongly depends on the location of the nodes in the initial triangular mesh. The generalization to 3D is however unclear and a second technique is investigated. This one aims at computing a near-optimal solution by using a *look-ahead tree* technique. The corresponding algorithm allows tuning the quality of the final mesh by choosing the depth of the tree as a parameter. This technique gives a promising way forward, especially as it is directly applicable in 3D.

The second subject concerns the development of a method that permits to compute, with respect to any prescribed tolerance, the extrema of Jacobian-based quantities defined on finite elements of any order and type. Applied to the Jacobian determinant, this method allows to assert the validity of any (curvi-)linear finite element. This method is also applied to a quality measure that quantifies the pointwise anisotropy of the elements. Besides being very attractive for hexahedral mesh generation, this method is especially useful for the analysis of curvilinear finite element meshes. It can moreover be an important component of optimization techniques for achieving robustness.

Résumé

Parmi les différents types de maillages d'éléments finis 3D, les maillages hexaédriques présentent des propriétés qui peuvent être extrêmement attrayantes telles que l'alignement avec les caractéristiques physiques ou un coût de calcul plus faible. Pour cette raison, et malgré la maturité des algorithmes de génération de maillages tétraédriques, la génération de maillage hexaédrique a toujours été un domaine de recherche prolifique. Pourtant, il n'existe pas encore d'algorithme robuste capable de générer des maillages totalement hexaédriques, conformes, sur des géométries arbitraires, tout en respectant un champ de taille imposée. Une difficulté qui subsiste est qu'il n'existe pas de méthode pour affirmer de manière robuste qu'un hexaèdre est valide. En effet, les hexaèdres linéaires peuvent être repliés sur eux-mêmes au même titre que peuvent l'être les tétraèdres courbes.

Cette thèse traite de deux sujets. Premièrement, deux techniques originales de génération de maillages quadrangulaires sont examinées avec le but de pouvoir les généraliser au problème 3D. Ces deux techniques prennent l'approche indirecte qui consiste à combiner des paires de triangles dans un maillage triangulaire initial d'entrée. La première technique, appelée *Blossom-Quad*, calcule la solution optimale de ce problème par rapport à un critère de qualité donné. Comme pour n'importe quelle autre méthode indirecte, la qualité de la solution dépend fortement de la localisation des nœuds dans le maillage triangulaire initial. La généralisation au problème 3D est toutefois incertaine et une seconde technique est examinée. Celle-ci vise à calculer une solution presque optimale en utilisant une technique d'*arbre de décision*. L'algorithme correspondant permet de commander la qualité du maillage final en choisissant la profondeur de l'arbre comme paramètre. Cette technique donne une voie à suivre prometteuse, particulièrement de par le fait qu'elle soit directement applicable au problème 3D.

Le second sujet traite du développement d'une méthode qui permet de calculer avec la précision voulue les valeurs extrêmes de quantités basées sur le jacobien qui sont par essence valables pour les éléments finis de n'importe quel type et n'importe quel ordre. Appliquée au déterminant jacobien, cette méthode, permet d'affirmer la validité des éléments finis linéaires ou courbes. Cette méthode est aussi appliquée à une mesure de qualité qui quantifie l'anisotropie

ponctuelle des éléments. En plus d'être très intéressante pour la génération de maillage hexaédrique, cette méthode est particulièrement utile pour l'analyse de maillages d'éléments finis courbes. Elle peut également être un composant important des techniques d'optimisation afin d'accroître leur robustesse.

Remerciements

Ecrire une thèse est une incroyable aventure que je n'oublierai jamais. Ce fut à la fois long et court, parsemé de hauts et de bas. Je voudrais remercier les nombreuses personnes qui ont contribué directement ou indirectement à l'écriture ou qui ont agrémenté toutes ces années de recherche qui ont abouti à ce manuscrit :

Il y a d'abord mon promoteur, Christophe Geuzaine, qui m'a non seulement proposé un excellent sujet de thèse dans lequel j'ai pu m'épanouir pleinement mais surtout qui m'a guidé tout au long de la thèse et ce, jusqu'aux derniers instants.

Il y a aussi mes collègues et anciens collègues ainsi que mon promoteur qui ont toujours su faire régner une super ambiance dans le service ACE.

La dernière ligne droite de la rédaction a été particulièrement intense. Je l'ai passée en grande partie retiré à la campagne, ce qui m'a aidé à tenir le coup. Je le dois à ma formidable mamie qui m'a fourni un espace de travail idéal et préparé de délicieux petits plats pendant ces nombreuses semaines.

Parmi mes amis, ma famille et mes collègues, beaucoup m'ont proposé de l'aide. Même si je n'ai pas toujours répondu à la proposition, ça fait toujours chaud au cœur de se sentir soutenu et je tiens aussi à les remercier.

Il y a de ces gens qu'on a envie de remercier pour leur présence, même lorsqu'ils n'ont pas de relations directes avec le présent travail. Je pense à :

Mes chers parents par qui tout a commencé. Ils m'ont conçus puis élevé de leur côté, chacun avec leurs moyens respectifs. Mon père m'a appris le sens du travail, ma mère à persévérer, et les deux m'ont toujours soutenu dans tout ce que j'ai entrepris.

Mon frère, toujours là pour m'apporter une aide précieuse lorsque j'en ai besoin.

Ma petite sœur adorée qui a apporté du soleil dans ma vie et qui grandi trop vite.

Mon très cher ami, Arnaud, que je n'arriverais décidément jamais à sur-

passer à l'escalade.

Enfin, comment ne pas mentionner la précieuse amitié que j'ai eu la chance de partager avec Sabine lors de mes études et au début du doctorat ? En plus des nombreux et chouettes moments passés en sa compagnie, celle-ci a été un véritable catalyseur à ma vie sociale au sein de l'université.

Contents

CHAPTER 1

Introduction

The Finite Element Method (FEM) is a powerful numerical technique that allows solving partial differential equations on complex geometries. Its origins go back to the 1950s with the appearance of the first digital computers. At this time, only large companies could afford such computers and, unsurprisingly, it is amongst aerospace companies that the method started to be developed. In the 1960s, the FEM began to be transferred to a wider range of engineering applications and, as a matter of fact, authors already acknowledged its power and generality at this time [1–5]. Since then, the method has continually gained fame and it is nowadays widely used in almost every field of engineering analysis, the domains of application being extremely vast: aeronautics, automotive, civil engineering, electrical engineering and electronics, biomechanics, soil mechanics, etc. It is considered that the number of finite element analyses performed every day around the world is of the order of the million [6], while FEM in industry weights several billion dollars per year [7]. The popularity of FEM in industry can be explained by the numerous benefits it brings: the possibility to analyze a design in detail and predict its performance and reliability, the possibility to optimize designs in order to reduce material usage, the reduction of costly physical prototyping and testing and an overall reduction of the time-to-market for new products.

A Finite Element analysis usually consists of four principal steps: geometry construction, mesh generation (the discretization of the geometry in small simple shapes), computation of the solution and post-processing. Although it may seem at first sight that the second step, mesh generation, is a trivial task, it is nothing of the sort. It is actually commonly admitted that it constitutes the major bottleneck of the process [6, 8–13]. This fact concerns above all complex industrial geometries and is caused by several factors [8, 14, 15]. On the one hand, block-structured meshes require to manually partition the ge-

ometry into blocks, which can take weeks or more for complex geometries. On the other hand, unstructured meshes are much more automatic but complex geometries require that the mesh generation parameters are fine-tuned, often in a trial-and-error process. Moreover, a significant number of geometrical models contain errors such as overlapping patches, gaps between patches or surface intersections, which require time-consuming and difficult-to-automate repairs.

Finite element meshes consist of a set of small simple shapes called finite elements that are the support for computing the solution field. Different types of elements exist: the most common elements are, in 2D, the triangle and the quadrangle, and in 3D, the tetrahedron, the hexahedron, the prism and the pyramid. Two main schools stand out: some users of the FEM prefer triangular and tetrahedral meshes for the availability of fast and robust generators while some users prefer all-quadrangular and all-hexahedral meshes which may offer better properties such as alignment with physical features or a lower computational cost. Quadrangular and especially hexahedral mesh generation are by far harder problems than their rival counterparts and have been the subject of numerous research works, as evidenced by the many PhD theses published over the years [16–24]. There is a real demand for automatic all-hexahedral mesh generators, and designing such a mesh generator that would work on arbitrarily complex geometries has been and still is the fantasy of many researchers...

Classical finite element meshes are composed of linear (straight-sided) elements over which the solution is interpolated using either linear or quadratic polynomials. Improving the accuracy of the solution can be achieved through the refinement of the mesh in what is referred to as the h-version of the FEM. Another approach is the p-version which consists in fixing the size of the elements but increasing the degree of the polynomial interpolation functions. It has been shown that the p-version may offer better convergence of the solution (see, *e.g.* [25]), provided also that the elements conform to curved boundaries, *i.e.* that high-order (curved) elements are used when necessary. Moreover, it has been established that a mix of the two approaches, called the hp-version could lead to a "superconvergence" [26]. There has thus been a frenzy in the 1980s to develop such methods but the complexity of the implementations hampered the momentum [27], in particular due to the difficulty to robustly produce high-order meshes that conform to curved boundaries. Consequently most of the current industrial-grade and commercial finite element packages are still based on at most second-order interpolation on straight-sided meshes.

There is currently a renewed interest for high-order mesh generation due to recent developments in the field of high-order finite element methods [28], such as discontinuous Galerkin [29, 30] or spectral [31, 32] methods. Those methods crucially rely on the availability of high-quality curvilinear meshes. Methods

for generating such meshes include smoothing [33] and elasticity analogies [34]. However, even when those methods work well, there is no guarantee that the resulting mesh will be valid. An alternative approach is to use optimization [35–38], which allows to directly maximize one or several quality criteria. These methods require the definition of proper quality measures of curvilinear meshes that can be used in the optimization formulations. As optimization is usually computationally expensive, it can realistically be applied only on small patches of elements. To this end, detecting elements that have to be optimized is essential.

Objective of this work

The objective of this thesis if twofold:

1. To design a quadrangular mesh generation algorithm that is automatic, produces high-quality quadrangles, is able to respect a prescribed input size field and is a good candidate for the extension to the 3D problem, while being reasonably fast.

2. To design efficient algorithms for asserting the validity and for measuring the quality of finite elements of any order and any type, for the purpose of generating and validating quadrangular/hexahedral as well as general, curved meshes.

Outline

The manuscript is organized as follows: In Chapter 2, we review state-of-the-art techniques for the two subjects we are interested in: (1) quadrangular and hexahedral mesh generation and (2) the validation of (curved) arbitrary order finite elements. In Chapter 3, we detail our contributions, which are appended in their original published form at the end of the thesis:

1. An algorithm that computes an optimal solution of the problem of combining triangles of an initial triangular mesh in order to produce a quadrangular mesh [39] (Appendix A);

2. A strategy that aims at efficiently computing a near-optimal solution of the same problem [40] (Appendix B);

3. A method for robustly and efficiently computing the extrema of the Jacobian determinant of curved, arbitrary order finite elements, and subsequently determine their validity [41] (Appendix C);

4. An extension of the above method to pyramids [42] (Appendix D);

5. A technique for computing the extrema of a Jacobian-based quality measure of curved, arbitrary order finite elements (Appendix E).

Finally, conclusions as well as our perspectives are presented in Chapter 4.

CHAPTER 2

State of the Art

The idea of discretizing space in order to compute numerical solutions of partial differential equations arose a hundred years ago: the first Finite Difference Methods (FDM) were developed on mechanical calculators for weather prediction and simple grids were used in order to discretize the atmosphere around the earth [14, 43]. The arrival of electronics and digital computers led to a dramatic increase in the use of numerical methods and the Finite Element Method (FEM) emerged in the 1950s. One significant advance that was brought by the FEM is more flexibility in the discretization of the geometrical domain: while FDM imposed the use of a grid that is aligned with the axes, the FEM relied on a mesh that can be totally unstructured. At the beginning, meshes were often created manually and in any case were usually quite simple: since mesh generation is a means to an end, users would commonly create their meshes in an ad-hoc manner for specific geometries. It is around the 1970s, as the need for accuracy of simulations increased, along with the size and complexity of the meshes, that mesh generation became a discipline on its own.

A finite element (FE) mesh consists of a set of small simple shapes called finite elements. There exists one 0D element, which is called a *node*, and one 1D element, called an *edge*. In 2D, there are two classical finite elements: the *triangle* and the *quadrangle* (a.k.a. quadrilateral). In 3D, *tetrahedra* (pyramids with a triangular basis) and *hexahedra* (cuboids) are the main finite elements but other elements also exist, *e.g. prisms* (a.k.a. wedges) or *pyramids* (with a quadrangular basis). Nodes, edges, triangles and tetrahedra are from a same family, the family of simplexes, so they can be referred to as the *simplicial* elements. Every element (except the node) is bounded by entities of one smaller dimension. For instance, a hexahedron is bounded by six quadrangular faces while a quadrangle is bounded by four edges. We consider that two elements are adjacent if they share an entity of one smaller dimension (an

edge for 2D elements and a face for 3D elements). Moreover, two elements are said to be connected if there exists a sequence of adjacent elements that starts by one of the two elements and ends by the other.

Any mesh has to respect a certain number of topological and geometrical constraints in order to be usable for a finite element simulation (see *e.g.* [44]):

1. *Topological validity*: Several conditions have to be satisfied for a mesh to be topologically valid. For instance, the elements cannot overlap, *i.e.* any point of the topological space can belong to at most one element. Moreover, the mesh must be connected which is true only if all the elements are pairwise connected. It is also required that no element is self-adjacent, *i.e.* an element cannot share a face with itself. Even if strictly speaking it has not to do with validity, doublets are also commonly forbidden because those elements, which share two or more faces, cannot possibly have an acceptable geometrical quality.

2. *Geometrical validity*: In addition to being topologically valid, geometrical validity requires that any geometrical point belongs to at most one element and that no element self-overlaps. The latter condition concerns quadrangular, hexahedral and curved elements.

3. *Topological and geometrical quality*: Even topologically and geometrically valid meshes might not lead to accurate FEM solutions. Notions of topological and geometrical qualities must then be introduced. However, the link between topological/geometrical quality and the accuracy of FEM solutions is often non trivial, as it depends on the mathematical structure of the underlying partial differential equations. Topological quality concerns above all quadrangular and hexahedral meshes and is not independent from geometrical quality: for instance, the best geometrical quality of a quadrangle usually corresponds to having every angle equal to $\pi/2$. This implies that every interior node of the corresponding mesh has a valence of 4 [1], which corresponds to a high topological quality.

Note that this is not an exhaustive list. For example, when using FEM on complex geometries (*e.g.* coming from CAD models), the mesh should closely approximate (be "close enough" to) its boundaries, and completely fill the volume. Such constraints are however not a concern for the present thesis.

[1] The valence of a node or an edge is a positive integer number. In a quadrangular mesh, it represents the number of quadrangles that touch a given node. In a hexahedral mesh, the valence represents the number of hexahedra that completely touch a given edge.

2.1 Quadrangular and hexahedral mesh generation

In contrast to simplicial mesh generation, which can be considered as mature since the 1990s [45–52] and for which robust algorithmic implementations are widely used [53–57], quadrangular and hexahedral mesh generation is still largely an open problem. Various reasons explain the demand for quadrangular/hexahedral meshes [58]:

- Empirical studies in structural analysis have shown that four to ten times less hexahedra are needed than tetrahedra in order to achieve a comparable accuracy, which implies less memory usage and less computational cost [59, 60].
- The linear tetrahedron is very stiff (mathematically, the eigenvalues of its stiffness matrix are larger than, *e.g.* the linear hexahedron). In elastic and elasto-plastic analyses, it implies that the linear tetrahedron locks in bending tests and using hexahedra [61] or quadratic tetrahedra [62] allows to overcome this drawback.
- Another well-known advantage of hexahedra is their effectiveness to capture anisotropic features. For instance, elements with high aspect ratios are required in boundary layers for Navier-Stokes computations and stretched hexahedra perform better than stretched tetrahedra. Hexahedra are also more appreciated than tetrahedra for anisotropically adapted meshes in presence of a directional flow [63]. Also, alignment of hexahedral elements with the material features of composite materials leads to better performance when performing structural dynamics simulations [58].
- The high-order spectral finite element method [32] makes use of the Gauss–Lobatto quadrature which leads to a diagonal mass matrix on quadrangles and hexahedra [64]. The spectral finite element method has been extended to triangles, at a higher computational cost than quadrangles, but not to tetrahedra [65]. Moreover, recent developments of the method on GPUs have highlighted promising scalability and throughput for hexahedra [66].

Before reviewing the main families of algorithms and techniques to generate such meshes, we highlight the main factors that explain the difficulties of quadrangular/hexahedral mesh generation.

2.1.1 Algorithm attributes

The quadrangular/hexahedral meshing algorithms have many characteristics that differentiate them from one another. It is common in the literature to

establish a list of them in the form of the desired attributes that the ideal mesh generation algorithm should have [11, 21, 23]. However, in this thesis, we prefer not to talk about an ideal algorithm, as some attributes are mutually conflicting (like the mesh size constraint and the mesh structure). Moreover, an algorithm can be well-adapted for certain classes of geometries or applications but not in other cases.

The following list is not exhaustive but tries to highlight the most common attributes that characterize quadrangular/hexahedral mesh generation methods:

- *Quad/Hex proportion*: It is usually implicitly understood that *quadrangular* and *hexahedral meshes* refer to meshes composed of only one type of element. However, while it is indeed possible to create such meshes in 2D, no general automatic algorithm generating fully-hexahedral meshes for arbitrary geometries currently exists. It is thus common to design algorithms to generate what are called *hex-dominant* meshes, which contain other types of elements (tetrahedra, prisms and pyramids) but for which the hexahedra are in the majority. By contrast, meshes that do not contain other types of elements can be explicitly referred to as *all-quadrangular* and *all-hexahedral* meshes. It is worth mentioning that there also exists *hybrid* meshes in which regions are meshed with structured hexahedral elements and other regions are meshed with tetrahedral elements. The connection between the different regions can be conformal or not (see the next attribute). At last, the term *mixed* mesh also exists to designate a mesh that contains several types of elements.

- *Conformity*: In a *conformal* mesh, two elements can either be unconnected or share a node, an edge or a face. This implies in particular that a hexahedron cannot share a face with a tetrahedron in a conformal manner. In a conformal hybrid mesh, it is thus necessary to use prisms and pyramids in order to make the transition between the two types of elements. A hexahedron that shares a face with two tetrahedra is one example of non-conformity. Other examples include a quadrangle that shares an edge with two smaller quadrangles or two hexahedra that share three nodes or more but no faces.

- *Geometry generality*: Ideally, a quadrangular/hexahedral meshing algorithm should be able to handle any kind of geometry. In practice, however, all-quadrangular/all-hexahedral algorithms have to neglect others attributes and there exist many *geometry-specific* mesh generators that can efficiently produce high-quality meshes for restricted classes of geometries.

- *Mesh structure*: It is common to separate meshes into two categories: *unstructured* and *structured* meshes. A mesh is said to be locally structured if in some neighborhood the nodes are such that they could be the nodes of a deformed regular (*e.g.* cartesian) grid. Among structured meshes, we distinguish three subcategories: *all-structured*, *block-structured* and *semi-structured* meshes. An all-structured mesh is a mesh that is structured everywhere. Every interior node of an all-structured quadrangular mesh has thus a valence 4. Similarly, each interior edge of an all-structured hexahedral mesh has a valence 4. Block-structured meshes consist of a small number of all-structured patches. Finally, we say that a mesh is semi-structured when the number of irregularities (valences different from 4) is limited. The main families of algorithms used to generate these three subcategories of structured meshes will be reviewed in the next section: mapping and submapping techniques for all-structured meshes, geometry partition methods for block-structured meshes and receding fronts, paving/plastering or grid-based methods for semi-structured meshes. Note that the structure is intimately linked to the notion of topological quality.

- *Mesh quality*: As briefly mentioned above, the FEM solution can be deteriorated by bad quality elements, which can also have an impact on the simulation time. Indeed, bad quality elements can *e.g.* reduce the stable time step of explicit time integration schemes or decrease the convergence rate of iterative schemes. The definition of what is a good-quality element is not well-established and different kinds of simulations can be influenced differently by the quality of elements, but it is commonly required that the angles of the quadrangles are as close as possible to $\pi/2$. In the same way, it is commonly required that the angles between two faces of an hexahedron are as close as possible to $\pi/2$. Note that the mesh structure is somehow correlated to the quality in the sense that high-quality meshes are very structured.

- *Element size control*: The smaller the size of the elements, the more accurate the solution. However, as computation time is directly linked to the number of elements, it is often desirable to generate coarser meshes in "less-important" regions and finer meshes in regions where high accuracy is needed, a process automated in h-adaptive analyses. The prescribed size field can often present strong gradings, which imposes strong constraints on the mesh structure (and geometrical and topological quality). Note that while some algorithms are not a priori designed to respect a prescribed size field, post-processing steps can always be applied to locally refine a mesh in conformal manner (*e.g.* [67, 68]).

- *Boundary mesh constraint*: There exist different strategies for meshing a multipart geometry. In the *top-down* strategy, the highest dimen-

sional mesh is generated first and lower dimensional meshes (surface, edge, node meshes) are extracted on the boundary of each part. On the contrary, in the *bottom-up* strategy, the boundaries of the geometry are meshed first, before creating the higher dimensional meshes. Some algorithms are not designed to respect a boundary mesh and are thus not well-suited for a bottom-up strategy.

- *Direct or indirect generation procedure*: Some algorithms generate first a triangular or a tetrahedral mesh as the foundation for generating the quadrangular/hexahedral meshes. Those algorithms are called *indirect* while the others are called *direct* algorithms.

- *Boundary sensitivity*: In some domains of application and particularly in fluid dynamics, the solution must be very accurately resolved close to the boundaries of the geometry. It follows that high-quality elements are strongly desirable near these boundaries. Methods that are designed to generate such meshes are called *boundary-sensitive*.

- *Orientation sensitivity*: The mesh produced by an *orientation-sensitive* mesh generator will depend on the orientation of the geometry in space, *e.g.* its (mis)alignment with coordinate axes. Geometries predominantly consisting of plane surfaces parallel to the coordinate axes can be handled well by such algorithms. In general, *orientation-insensitive* algorithms are however more desirable.

- *Tolerant to bad geometry*: Complex geometries that go from the design step to the meshing step often present bad geometry definitions such as overlapping patches, gaps between patches or surface intersections. Repairing those geometries is a very time-consuming, often manual task. Handling gracefully bad geometries is an important attribute for industrial mesh generators.

- *Computational cost*: It is always advantageous that an algorithm is as fast as possible. This is particularly true for mesh generation for which parameters must often be tuned in a trial-and-error manner. On a personal computer, automatic tetrahedral mesh generators are nowadays capable of generating more than one million tetrahedra per minute [57]. Large industrial meshes are generated on supercomputers and can easily contain more than 1 billion tetrahedral elements. It is difficult to establish an acceptable computation cost for quadrangular/hexahedral mesh algorithms. This would depend on the type and the quality of the mesh, as well as the class of geometries the algorithm can handle, etc. However, it is reasonable to say that for an industrial application, the computational complexity of meshing algorithms should be as close to possible to linear.

To conclude, let us mention that mesh generators can have different degrees of automation. As mentioned above, for arbitrary complex geometries, fully-automatic hexahedral mesh generators do not exist. Popular industrial approaches are based on a mostly manual partitioning of the geometry, which requires a lot of human time and experience.

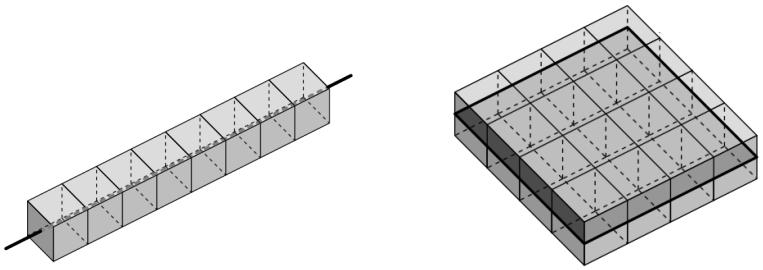
2.1.2 Conformal all-quad/all-hex meshing constraints

In addition to the topological and geometrical constraints common to all type of finite element meshes, all-quadrangular and all-hexahedral meshes have other constraints that make them particularly difficult to generate as well as less flexible for *e.g.* adaptation.

To better understand these constraints, it is useful to consider the dual graph of the mesh, which is defined as the graph whose vertices correspond to mesh elements and whose edges join pairs of adjacent elements. Duals of quadrangular and hexahedral meshes have a particular topological structure that is called *Spatial Twist Continuum* (STC) [69]. This structure is described by two entities (see Fig. 2.1):

- *Chords*: A quadrangle can be seen as two pairs of opposing edges and a hexahedron can be seen as three pairs of opposing faces. This involves in the dual graph that edges incident to a vertex can be uniquely paired. We can thus give this definition to a chord: it is a sequence of connected dual edges for which each two consecutive edges correspond to opposing edges/faces of the mesh. In other words, a chord represents a stack of quadrangles or hexahedra. A chord can either be a closed loop or start and terminate at the boundary. As a consequence, quadrangular (resp. hexahedral) meshes must have an even number of edges (resp. faces) on the boundary.

- *Sheets*: Two adjacent hexahedra have 5 chords running through them. One is common to the two hexahedra and the other four can be uniquely paired such that the paired chords are locally parallel. As a consequence, two chords that are locally close (*i.e.* running through the same hexahedron or two adjacent hexahedra) are either locally parallel or locally perpendicular. A sheet can then be defined as follows: it is the surface that contains a collection of chords for which each set of locally close chords are perpendicular to a foreign chord. In the primal mesh, it corresponds to a layer of hexahedra. Similarly to chords, a sheet can be a closed surface or its boundaries have to coincide with the geometry boundary. A sheet can self-intersect and the intersection of two sheets or two parts of a sheet is a chord.



(a) (b)

Figure 2.1: Dual entities of the hexahedral mesh: (a) A chord which corresponds to a stack of hexahedra and (b) a sheet which corresponds to a layer of hexahedra. (Pictures from [23].)

Any topological modification applied on a mesh can be seen from the STC point of view. For quadrangular meshes, a collapse for example corresponds to (1) cutting two chords which intersect at their intersection and (2) reconnecting the parts pairwise in a way that they don't intersect anymore (see Figure 2.2). Similarly, removing a doublet consists in reducing the number of intersections between two chords by one. Because of the sheets, there is no such local modification in hexahedral meshes. A modification that affects a sheet has to propagate along at least one chord included in that sheet. For instance, the generalization of the 2D collapsing is changing the topology of the STC so that an intersection between two sheets is removed. In other words, collapsing a given hexahedron imposes to collapse all the hexahedra that are run through by the corresponding chord.

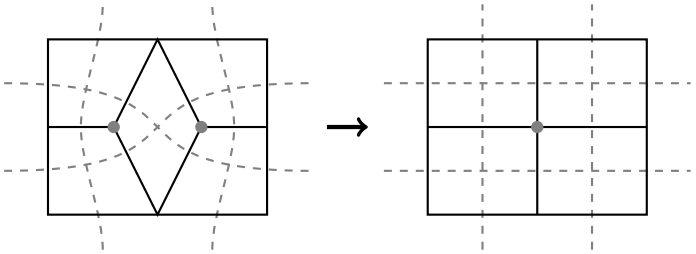
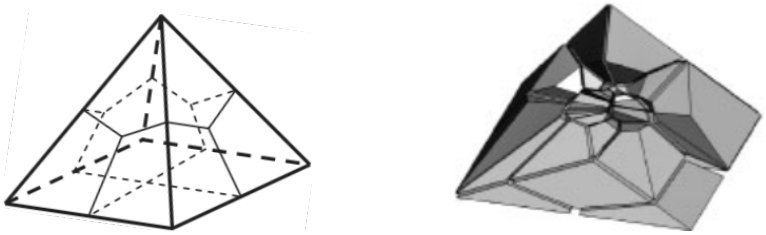


Figure 2.2: The collapse operation consists in removing a quadrangle and merging to opposing nodes in order to close the created hole. In the STC point of view, it consists in modifying two intersecting chords.

This explains two things: (1) The topological post-processing performed on a all-hexahedral mesh is a non-local process and it offers therefore less possibilities and (2) more generally, the problem of generating an all-hexahedral mesh is much more constrained and therefore much harder than the problem of generating other types of meshes.

As an illustration, let us consider a geometry that is topologically a n-holed torus. It is possible to cut each of the n-handles in turn by non-self-intersecting topological disks bounded by an even cycle of edges in order to topologically reduce the geometry to a sphere [70], which admits an all-hexahedral mesh provided that the number of faces on its boundary is even. However, this does not ensure that it is possible to generate a good-quality (or even a valid) mesh. One famous example is Schneiders' pyramid for which every face is discretized with quadrangles in an elementary manner [71]. There are four quadrangles on the base and three quadrangles on each triangular faces (see Figure 2.3). A reasonable number of hexahedra in order to fill the volume would be 5 or 6 (such that the size of each element remains close to the size at the boundary). Because of the top corner, that has four adjacent edges, such a solution is however impossible. The minimal solution in term of the number of sheets implies 20 hexahedra [72]. This solution contains however many doublets and does not meet the topological validity constraints (and geometrical quality) required for a finite element simulation. Up to now, two valid solutions have been published, which respectively use 118 and 88 hexahedra [73, 74]. However, the resulting elements have poor scaled Jacobian determinant and the meshes are thus of poor quality, which restricts their interest for industrial applications.



(a) (b)

Figure 2.3: (a) Schneiders' pyramid and (b) a cut of the 88-elements solution proposed by Yamakawe and Shimada. (Pictures from [74].)

In the following sections we review a non-exhaustive list of the state-of-the-art techniques for generating quadrangular and hexahedral meshes (further details can be found in [58, 75, 76]).

2.1.3 Core methods

These methods are generally easy to implement and constitute in some way the basis of quadrangular and hexahedral mesh generation. They only apply to elementary geometrical shapes, so they are used either to mesh simple geometries or in the partition techniques (Sect. 2.1.4). It is important to point out that all of these core methods produce meshes that are strongly constrained.

For instance, mapping methods require that the number of edges/faces must be the same on opposite sides of the domain.

Mapping

Mapping is the natural technique for generating an all-structured mesh. It consists in deforming a grid defined on a computational domain such that it fits into the physical domain, *i.e.* the geometry. This can be made by an explicit mapping or an implicit mapping, the former being generally more efficient than the latter. There is one requirement for the geometry to be mappable: it must be topologically equivalent to a quadrilateral or a cuboid. In other words, the user must determine four logical sides from a surface geometry or six logical quadrilateral faces from a volume. The quality of the mesh that is obtained depends on the shape of the geometry: the more the resemblance with a rectangle or a rectangular cuboid, the better the quality.

The *Transfinite Interpolation* (TFI) method [77] is one of the most popular algorithm for producing all-structured meshes [78] (Fig. 2.4). The mapping is defined by an explicit algebraic expression. More precisely, it is defined through interpolation functions that can have different forms: linear, square, cubic, Hermite or Bernstein polynomials. The TFI method is computationally efficient, in term of both memory and time, and is easy to implement. Some improvements extend the applicability of the TFI method [79, 80].

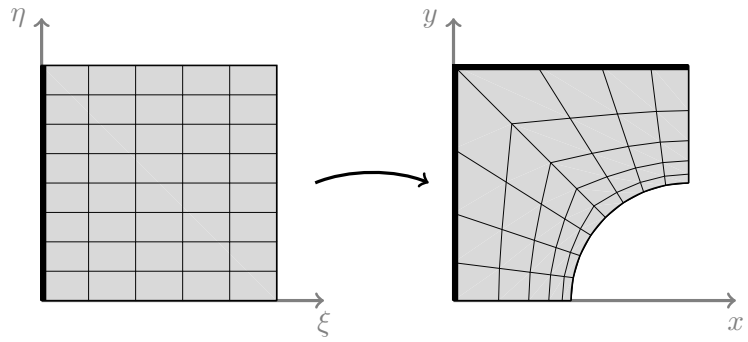


Figure 2.4: Among the various algorithms based on mapping, the Transfinite Interpolation method is one of the most popular. The mesh of the computational domain (left) can be mapped to the physical domain (right) if four logical sides are identified. Here, the two physical border edges drawn in thick black on the right mesh form one unique logical side.

Another family of mappings includes explicit conformal mappings [81–83]. By definition, those mappings preserve the local angles. The grid defined on the computational domain is orthogonal and the angle preservation im-

plies that the resulting mesh (composed of straight-sided elements) is near-orthogonal. Explicit conformal mappings have been largely used for meshing outer domains, principally for fluid simulations. It was really appreciated in FDM for which a (near-)orthogonal grid avoids to add cross-derivative terms for the resolution of the equations. In contrast with the TFI method, a conformal mapping is inherently a 2D technique.

PDE-based methods [84, 85] have also been widely used to produce all-structured meshes. The mapping is defined by an implicit PDE equation that can be elliptic, hyperbolic or parabolic. Elliptic methods have the great advantage to produce smoother meshes than the TFI method but are more computationally expensive. Hyperbolic methods are useful to mesh outer domains by propagating an initial boundary mesh. They provide grid orthogonality and are faster than elliptic methods. They however propagate the singularities of the boundary. Parabolic methods combine some of the advantages of both elliptic and hyperbolic methods. In particular, they provide smooth meshes while being computationally efficient.

Primitives

Mesh generation based on primitives consists in meshing simple geometrical shapes with predetermined templates [86]. Examples of simple geometrical shapes are polygons, circles, boxes, tetrahedra, spheres, cylinders, etc. The template of the tetrahedral primitive, for instance, consists in dividing a tetrahedral shape into four hexahedral blocks. Those blocks are then meshed by using a mapping technique. The geometrical shapes do not need to be straight-sided, *e.g.* the template of the tetrahedral shape can be used to mesh an octant of an sphere (Fig. 2.5). Algorithms based on primitives are fast and easy to implement and are thus often included in mesh generators. They also usually produce good-quality meshes. An interesting method for meshing some primitives is the midpoint subdivision [87].

The cylinder primitive, that is created by ‘‘sweeping’’ a quadrangular mesh from one end to the other, has been extended to more advanced algorithms that are presented in the next paragraph.

Sweeping

The sweeping technique consists in creating a 3D mesh by projecting the mesh of a surface along a specified trajectory. In one respect, sweeping is to meshing what extrusion is to CAD modeling. The geometries must possess the following characteristics (see [88] for details):

- Every boundary surface is either a *source* surface, a *target* surface or a *linking* surface (see Fig. 2.6). The source surface and the target surface

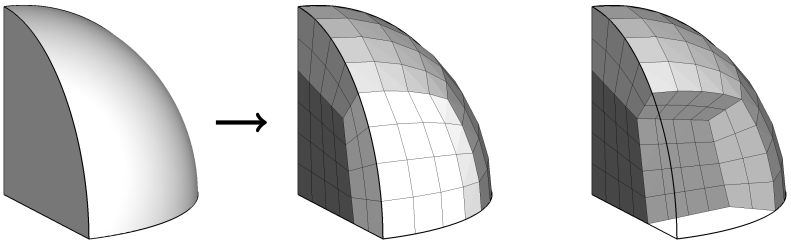
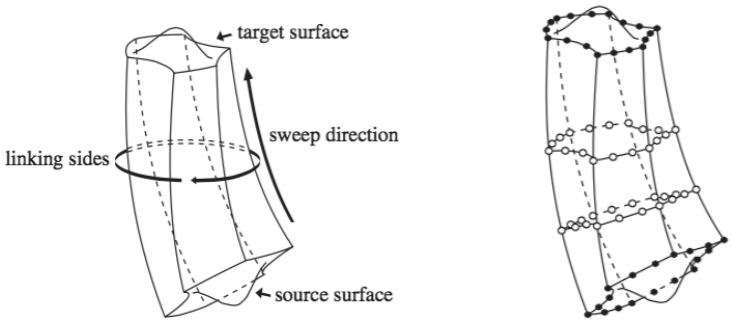


Figure 2.5: Techniques based on primitives allow meshing simple geometries such that this octant of a sphere (left). The template of the tetrahedral shape can be applied (middle). It consists of four hexahedral blocks that are meshed using a mapping technique. The image on the right shows the same mesh for which the front block has been hidden.

are unique.

- The source surface and the target surface (also called *cap* surfaces) must be topologically the same. In particular, they must contain the same number of logical sides and the same number of holes.
- The linking surfaces must be bounded by 4 logical sides.



(a) (b)

Figure 2.6: Topological shape of a sweepable volume. The source and target surfaces must be unique and the linking surfaces must be topologically equivalent to a quadrilateral. (Pictures from [89].)

The volume mesh can be generated in four distinct steps: (1) generation of an all-structured quadrangular mesh on the linking surfaces, (2) generation of the mesh on the source surface, (3) projection of this mesh onto the target surface and (4) generation of the inner nodes and volume elements. This results in a layered mesh for which the layers have the following characteristics: they are perpendicular to the sweep direction, they are all topologically the same and correspond to an extrusion of the source surface mesh. Due to

this particularity, they are often called 2.5D meshes and the geometries, 2.5D geometries. The meshes of the linking surfaces are generated using a mapping technique. Many sweeping methods consider the nodes of the linking surfaces as a stack of *bounding loops* for the construction of the interior nodes. The mesh of the source surface can be of any type (triangular, quadrangular or mixed) and can be generated using any existing technique for generating a mesh on a surface. The challenging steps of the method are the two remaining steps: projection of the source mesh to the target surface and location of the interior nodes inside the volume. The mesh of the target surface must have the same connectivity that the mesh of the source surface. Several methods can be employed to construct it:

- *Smoothing*: The method consists in constructing a mesh on the target surface of the same connectivity that the mesh on the source surface with an initial node placement that may not necessarily be good. This can be done by computing *e.g.* an affine mapping. Then, a smoothing operation is applied in order to improve the quality [90]. This approach can give really good results on a various types of geometries but smoothing routines are typically slow and not well-suited for large meshes.
- *Least-squares approximation of an affine mapping*: This method requires that the two surfaces have a parameterization. The idea is to compute an affine mapping that will transform the nodes of the source surface to the target surface in the parametric space. Since the boundary nodes of the target surface are already known (after the linking surfaces have been meshed), the mapping would be the mapping between the boundary nodes of the two surfaces. This mapping is not affine in the general case, thus the affine mapping is computed in the least-square sense [89].

Generation of the inner nodes can be done by one the following methods:

- *Geometric placement*: Similarly to a frontal method, this method sequentially creates the hexahedra of each layer [91]. A hexahedron is created only if seven nodes are known and the last node is placed such that the three faces of the new element touched by the node are planar. This limits the distortion of the final mesh and works well for specific blocky geometries.
- *Layer smoothing*: As for generating the mesh on the target surface, the location of the interior nodes can be computed by smoothing the 2D mesh of each layer [92]. This results in planar node layers which is not adapted for geometries with curvatures on the source and target surfaces.
- *Faceted node projections* [93]: This method uses a background triangular mesh in order to locate the new nodes. This background mesh is

first generated on the source surface by tessellating the boundary nodes. Then it is elevated to each layer of nodes that has to be created. Each new interior node is computed using the barycentric coordinate of the triangle which contains it. At the end, the volume elements are created by connecting the nodes. An offset distance is considered in order to take into account the curvature of the source and target surfaces, such that the resulting mesh is smooth. Inverted elements can however be introduced in the background mesh if the geometry presents distorted holes.

- *Least-square approximation of an affine mapping*: The method consists in computing the interior nodes of a given layer as an affine mapping of the nodes from the source surface. Some geometries require only translation, rotation, and scaling of layer meshes. In this case, the mapping between two bounding loops of the linking surfaces is affine and the interior nodes are placed according to this exact affine mapping. For more complex sweeping, an affine mapping may not exist but an affine mapping can still be computed in the least-square sense [92]. In this case, an error exists between the computed position and the position that would be computed by using the actual mapping of the bounding loops. Note that a functional has to be minimized which leads to solve a linear system that can sometimes be singular. The minimization of another equivalent functional [94, 95] can overcome this drawback while improving the matching respect of the curvature of the source and target surfaces. The node location can again be improved in different ways [96–98].

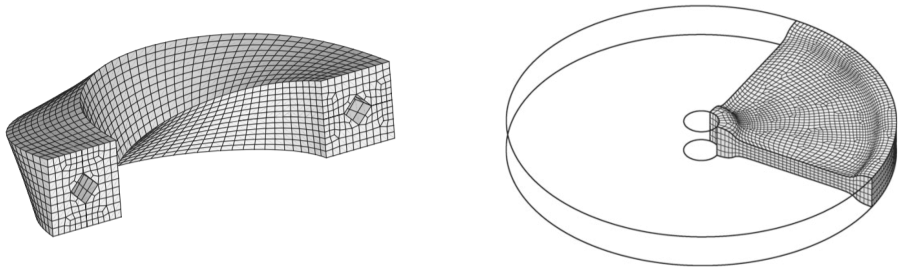


Figure 2.7: Sweeping techniques can handle various type of geometries as, for example, geometries with twisted and curved sweep path (left) or high curvature of the source and target surfaces (right). (Pictures from [89, 95].)

Figure 2.7 presents two geometries that have been meshed using the sweeping technique. Extensions to sweeping techniques that allow to handle geometries with multiple source and target surfaces are presented in the next section.

2.1.4 Geometry partition methods

Geometry partition is an important class of all-hexahedral mesh generators. Formerly, the term geometry decomposition was used in order to designate such methods but some authors began to use the more explicit term *partition*, which we also prefer to use. The technique consists in partitioning the geometry into subdomains of simple shape that can be meshed by one of the core methods presented in the previous section. Block-structured meshes are typically obtained by meshing the subdomains with a mapping technique. Those meshes are usually of very good quality due to their structure. However, the conformity of the mesh between the adjacent subdomains implies that the constraints on the mesh size—peculiar to the core methods—propagate through the whole geometry. The number of elements, and thus the discretization, cannot be chosen in a particular region of the geometry without having consequences in other regions. The control of the element size is thus highly limited. Specifically, refining the mesh in a region of interest will usually also refine the mesh in other unwanted parts of the geometry. Theoretically, any kind of geometry can be partitioned in order to mesh the subdomains with the core methods. However, automatic partition methods still struggle to handle complex geometries. For this reason, the partitioning of complex geometries is a human-time-consuming manual process.

Those methods are not designed to respect a given boundary mesh and are thus not suited for a bottom-up strategy. They are moreover all direct algorithms.

Submapping

The submapping technique produces all-structured meshes on geometries that have not necessarily the shape of a quadrilateral or a cuboid [99, 100]. Even so, the geometries should have angles that are relatively close to 90° in order for the generated mesh to be of good quality. Moreover, for 3D geometries, the faces must be mappable or submappable. This technique is well-adapted to blocky geometries and is popular in mechanical simulations. The quality of the generated mesh can be very good if the geometry lends itself to the submapping, both inside and near the boundary. Figure 2.8 shows a mesh obtained by the submapping technique.

In 2D, the submapping algorithm relies on a classification of the vertices of the surface to mesh. This classification allows to consider a representation of the surface for which the edges are vertical or horizontal (see Figure 2.9). This representation is called the computational domain. (In 3D, the computational domain is sometimes referred to as the polycube.) The edges are then classified according to their direction in this computational domain. In order to be all-structured, the final mesh must verify the property that the number of

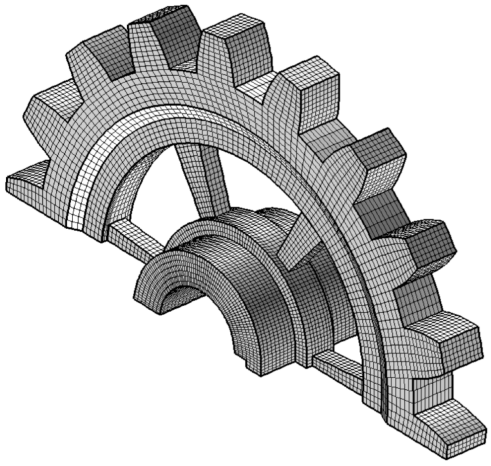


Figure 2.8: Submapping of a gear. (Picture from [101].)

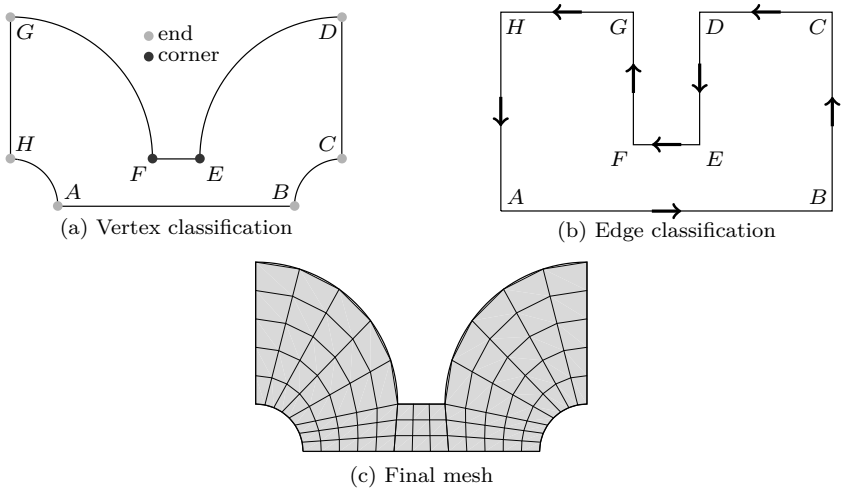
intervals on edges going upward (resp. from left to right) is equal to the number of intervals on edges going downward (resp. from right to left). This constraint is enforced by solving an integer linear problem (ILP). Then, virtual edges are created that partition the geometry into mappable regions. Those regions are meshed using a mapping technique and the final mesh is taken as the union of those meshed regions. The submapping of 3D geometries relies on the same approach: (1) classification of vertices, edges and faces, (2) resolution of a ILP for assigning the intervals on the edges, (3) partition of the geometry into cuboidal regions using virtual faces and (4) meshing of the regions using a mapping technique.

References [101, 102] improve the vertex classification and the interval assignment and extend the applicability of the method to surfaces and volumes with holes and voids. The proposed method automatically reduces multiply connected geometries into simply connected geometries by virtually connecting the holes or the voids to the exterior boundary.

Let us mention [103] in which it is proposed to generate all-structured meshes on general geometries without partitioning. This improves stability in comparison to other existing submapping methods. An initial manual partition is however needed, making the method not fully automatic. Moreover, the position of the inner nodes is computed by solving a sparse linear system.

Multisweeping and multiaxis sweeping

Geometries on which one-to-one sweeping can be applied must have only one source and one target surface (see Section 2.1.3). More complex geometries exist that can accept a swept mesh: geometries with multiple source sur-



(a) Vertex classification

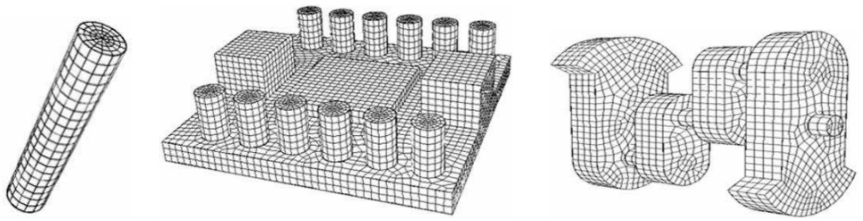
(b) Edge classification

(c) Final mesh

Figure 2.9: The submapping relies on a classification of the geometric entities: vertices, edges and faces (in 3D). The classification of the vertices (a) is function of the angle defined between their two adjacent edges. There are four classifications: end, side, corner or reversal. From this information, the computational domain (b) is constructed. The edges are then classified in function of their direction. The final mesh (c) is obtained after dividing the geometry into quadrilateral regions and applying the mapping technique to each of them.

faces but a single target surface in which case the meshing algorithm is called *many-to-one* sweeping, and geometries with multiple source and multiple target surfaces for which the algorithms are called *many-to-many* sweeping (see Figure 2.10). A method for detecting whether a geometry is sweepable, based on geometric and topological informations, has been developed in [88]. Additionally, the method is able to detect the source, target and linking surfaces of sweepable geometries. The linking surfaces must be meshed with a mapping or submapping technique. A method for assigning the intervals of sweepable geometry is described in [104].

A first approach to multisweeping was described in [91]. This method is the only one that does not perform any partition. The source and linking surfaces are meshed and a layer index is associated to each source surface according to its position in the linking surface. The meshing of the interior is done frontally, beginning from the initial source surface(s) and by the geometric placement technique described in Section 2.1.3. Each time a source surface is encountered, its nodes are added to the front. Each time a target surface is encountered, the nodes that touch it are used to produce the mesh of this surface and are removed from the front. This method can only work well for simple blocky geometries but is not suited to more complex sweepings. The



(one-to-one) (many-to-one) (many-to-many)

Figure 2.10: Different meshes obtained with a sweeping or a multisweeping technique. (Pictures from [21].)

reason is that the front moves forward without knowing where it goes.

In order to overcome this drawback, current approaches are based on real and virtual partitions. The difficult part for doing this is the "imprinting" process. Source and target surfaces do not have the same topology in general. The imprinting process consists in subdividing source surfaces and/or target surfaces such that they topologically match. Two approaches for many-to-many sweeping have been developed:

- *Virtual partition into barrels*: This approach consists in virtually partitioning the geometry into "barrels", *i.e.* sub-volumes that have a single source area and a single target area and can be meshed using a one-to-one sweeping technique [96]. The process begins by meshing the source and linking surfaces. Then the imprinting is performed and the barrels are formed. It is worth mentioning that barrels can share a source/target area; they can thus be grouped in stacks of barrels that connect source and target surfaces of the geometry. The next step is the projection of the source surfaces to each source and target areas of the barrels. Finally, the interior of the barrels is meshed separately and the final mesh is the combination of the elements in all the barrels. Note that the linking areas are predetermined (by the "ribs") such that the conformity is ensured between adjacent barrels. Reference [105] introduces Boolean operations for the imprinting operation such that more configurations and more complex geometries can be dealt with.

- *Partition into many-to-one sweepable volumes*: The goal of this method is to partition the geometry into sub-volumes that have a single target surface in order to be subsequently meshed by any existing many-to-one sweeping technique [97, 106]. The method uses its own mesh size, different from the size prescribed by the user because meshes that are created have for unique purpose to produce the partition. The linking surfaces are meshed with a mapping or a submapping technique. Using this information, the bounding loops of the target surfaces are back-

propagated to the source surfaces. The bounding loops of the target surfaces are then imprinted on the source surfaces in order for them to topologically match the target surfaces. Finally, triangular meshes are created between the source and target surfaces inside the geometry to represent the partition. An improvement of this method, which increase the robustness of the computed imprints in particular, has been presented in [107].

A many-to-one method has been described in [108] and a multiaxis sweeping algorithm has been presented in [109].

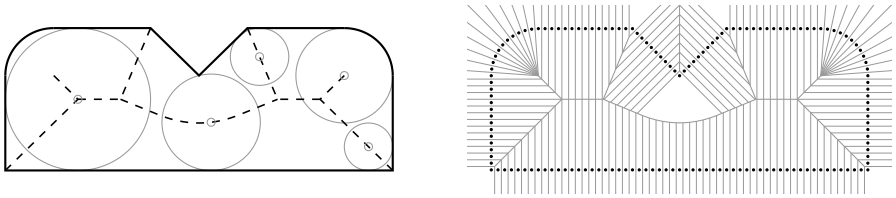
Medial axis

The *medial axis* is the set of points of a geometry for which there are more than one closest point on the boundary of the geometry. A more convenient definition can be given. Let us say that a disk (in 2D) or a ball (in 3D) is *maximal inscribed* to the geometry if it is entirely contained inside and if it is tangent to the boundary at two or more points. Thus, amongst all the disks/balls included inside the geometry that are tangent to a given point of the boundary, the maximal inscribed disk/ball is the one that has the larger radius. The medial axis can then be defined as the location of the center of all the maximal inscribed disks/balls, which is illustrated in Figure 2.11(a). Note that in 3D, the medial axis is also called *medial surface*.

The medial axis is somehow the skeleton of the geometry and it equivalently represents the geometry when in combination with the radius of the maximal inscribed discs/balls. This is known as the medial axis transform (MAT). The medial axis can be used to partition any given geometry. The first step is to construct it. The medial axis is usually extracted from a Voronoi diagram [2] since the medial axis is approximately a subset of the Voronoi diagram. First, the boundary of the geometry is sampled, which gives a discretization of the boundary. Then the Voronoi diagram is computed (see Figure 2.11 (b)). Finally, Voronoi edges and faces intersecting the geometry and entities located outside the geometry are removed. The remaining entities constitute an approximation of the medial axis. Details on how to construct the medial axis can be found in [110–112]. A drawback of the method is that the sampling of the geometry has to be dense in order to obtain a topologically correct medial axis and a robust method is still lacking for general 3D geometries. Moreover, in some situations, the medial axis can be degenerate: for instance,

[2] The Voronoi diagram of a given set of points is a partition of the space into Voronoi cells. To every point is associated a Voronoi cell and the Voronoi cell of a point P corresponds to the region of the space in which P is the closest point amongst the set. A Voronoi diagram is shown in Figure 2.11 (b).

a medial axis face degenerates to an edge in a square prism and the geometry of Figure 2.11 has two degenerate medial axis vertices.



(a) (b)

Figure 2.11: A geometry can be equivalently described by the medial axis transform (MAT), *i.e.* the combination of the medial axis and the radius of the maximal inscribed disks/balls. (a) A geometry (in black solid) and the corresponding medial axis (in black dashed). Some maximal inscribed disks are represented in gray. The medial axis can be extracted from the Voronoi diagram. (b) The Voronoi diagram (in gray) corresponding to the given sampling of the boundary (black dots).

The medial axis can be used for many purposes. A first use is the partition of the geometry into sub-volumes [113–115]. In this technique, the sub-volumes can be meshed using a midpoint subdivision technique proposed by [87]. This approach usually produces poor quality meshes in regions where the medial axis is degenerate. This can be overcome by computing a subdivision based on the embedded Voronoi diagram [116]. Even if it is closely related to the medial axis, the embedded Voronoi diagram is much easier to compute. The sub-volumes are most of the time sweepable in which case an all-hexahedral mesh can be generated but some sub-volumes that cannot be meshed with only hexahedra can be created.

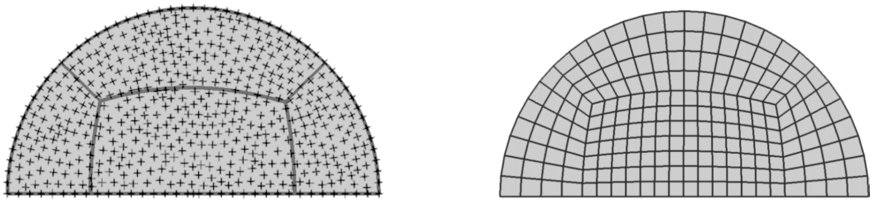
The medial axis can also be used for detecting thin regions [117–119]. This information can be useful for partitioning and reducing the number of degrees of freedom for the mesh generation. The detection can be done by comparing the radii of the maximal inscribed discs of the boundary faces (which give approximately the lateral dimensions) with respect to the radii of the maximal inscribed balls of the volume (which give approximately the thickness).

Another approach is to generate a mesh on the medial axis, which is subsequently extruded to the boundaries of the geometry. This approach provides all-quad/hex-dominant meshes [110]. However, a large number of non-hexahedral elements are generated on complex geometries and there even remain elements with 7 vertices (hexahedra with one collapsed edge), which are usually not handled by solvers. An improved algorithm is thus presented in [120].

A last use of the medial axis is for analyzing the geometry and assisting the user in the manual partition process [121]. The suggested sub-volumes are sweepable and an all-hexahedral mesh can be generated.

Cross-field based methods

Some recent works take advantage of the computation of a cross-field. The cross-field can be viewed as the preferred orientations of the quadrangles/hexahedra defined at each point of the domain, see Figure 2.12(a). The approach takes its origin from the graphics community where such fields have been used for visual effects such as anisotropic shading or texture synthesis. On surfaces cross-fields have also been used to compute global parameterizations from input triangular meshes [122–125]. These parameterizations can be aligned as desired, *e.g.* in the direction of the main curvatures, and inherently provide all-structured quadrangular meshes. However, this approach is not able to conform to prescribed boundary meshes or respect a given prescribed element size field. Another use of cross-fields is partitioning the domain into four-sided regions [126, 127]. Indeed, it has been demonstrated that singularities are the crucial characteristic features of cross-fields [128], which corresponds to irregular nodes (nodes for which the valence is not 4). A partitioning of the domain can thus be constructed using this information, see Figure 2.12. The cross-fields can be computed in different manners, by solving an elliptic PDE [127] or by using a fast-marching algorithm [126]. An advantage of the method is that it tries to compute a good position for the singularities and that the boundaries of the partitions are curved.



(a) (b)

Figure 2.12: A cross-field can be used for partitioning. (a) The cross-field represented by crosses in which two singularities are visible and the corresponding partition into four sub-volumes. (b) The resulting mesh. (Pictures from [127].)

Computation of a global parameterization in 3D has been proposed in [129, 130]. The cross-field is first computed on 2D surfaces before being is extended inside the volume. Other techniques relies on the minimization of an energy function [131, 132] and on the construction of a singularity graph for partitioning the geometry [133].

2.1.5 Octree-based methods

Octree-based (or grid-based) methods [67, 134] are able to produce all-hexahedral meshes on arbitrary geometries in a fully automatic manner, at the cost of degraded element quality near boundaries. The generation is done in a top-down strategy: an interior mesh is generated from a cartesian grid or an quadtree/octree if a mesh size field is prescribed. Elements that are not completely located inside the volume are not generated. Then, the boundary of this mesh is projected or adapted to the boundary of the geometry. A mesh obtained with a grid-based method is shown in Figure 2.13. When quadtrees/octrees are used instead of a cartesian grid, the mesh can be left non-conformal or it can be made conformal by using transitions templates as shown in Figure 2.14. The set of templates presented in [67] has been further extended in order to improve the adaptation of the mesh to the geometry [68].

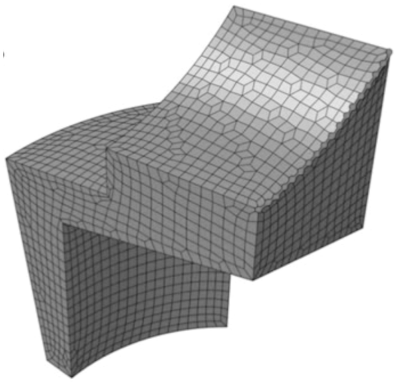
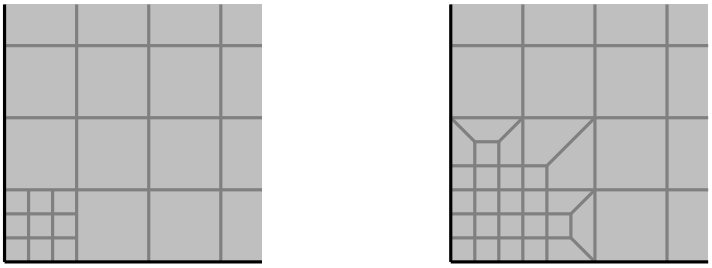


Figure 2.13: Meshes produced by a grid-based technique are all-structured inside the volume and unstructured at the boundaries of the geometry. (Picture from [135].)

Many extensions have been proposed. In references [136, 137], the method is adapted to handle geometries defined by volumetric data, which is a common type of geometry in bio-medical engineering. Reference [138] proposes a method to handle geometries with heterogeneous materials. For those geometries, the whole volume is still meshed at the same time but the interfaces between different materials are subsequently recovered. This method is improved in [139]. The capture of embedded features inside the mesh is a related problem and a method is presented in [140].

2.1.6 Direct advancing front methods

In order to be able to respect an initial boundary mesh, several advancing front methods have been designed. They all need as the minimal information an



(a) (b)

Figure 2.14: In 2D, in order to tune the size of the elements, a quadtree can be used instead of a cartesian grid. The mesh is initially non-conformal (a) but it can be made conformal (b) by applying transition templates.

initial boundary mesh. An advantage of those techniques is that the resulting mesh is usually of high quality and well aligned near the boundary.

The *paving* algorithm has been proposed for creating quadrangular meshes [141, 142]. It sequentially adds quadrangles starting from the boundary and manages fronts. Fronts thus propagate to the interior of the geometry and collide, see Figure 2.15. Collisions between fronts are the main difficulty of the method. They have to be well handled in order to construct a good quality mesh and such that the voids that are left can be meshed with only quadrangles if necessary. The method has been improved in [143] and the problem of meshing 3D surfaces has been treated in [144].

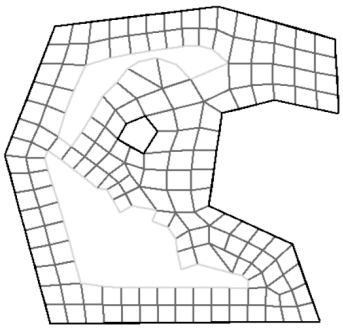
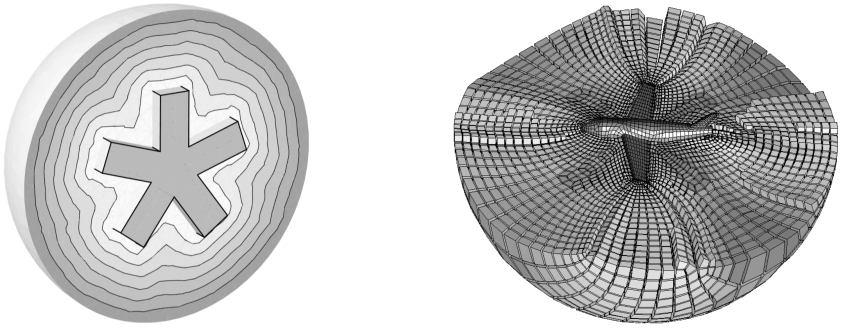


Figure 2.15: The paving algorithm in action. The light gray edges are part of the two fronts that remain after that two collisions occurred. (Picture from [143].)

The generalization of paving for hexahedral meshes is the *plastering* algorithm [16, 145, 146]. It starts from a quadrangular mesh on the boundary and creates hexahedra the same way than for the paving algorithm. However,

due to the additional constraints inherent to all-hexahedral meshing (see Section 2.1.2), this method can create inner voids that cannot be further meshed with hexahedra leading to the creation of a hex-dominant mesh. It is the price to pay for respecting the initial boundary mesh. The *unconstrained plastering* (and in 2D, the unconstrained paving) [65, 147, 148] has been designed in order to obtain more chances of success. The method releases itself from the boundary mesh constraint. A background tetrahedral mesh is used in order to compute successive fronts. Those fronts are computed independently of any hexahedra; it is with the intersection of the fronts that the hexahedral elements are defined. At the end, inner voids that cannot be hexahedralized may still remain.

Another disadvantage of those techniques is that bad elements are usually created at the meeting of fronts. The main reason is that fronts are propagated towards each other without anticipating the collisions. A recent method called the *receding front* method [149] has been designed in order to overcome this drawback. It only works for exterior domains, however. The novelty of this technique is to precompute the fronts in order to obtain a smooth transition between the inner surface and the outer surface. This is done by computing the level set of both surfaces. By combining them, a field that smoothly goes from 0 on the outer surface to 1 on the inner surface is obtained. The fronts are extracted from this field, see Figure 2.16. Then, the hexahedral mesh is constructed layer by layer from the quadrangular mesh of the inner surface to the outer surface. An all-hexahedral mesh is obtained. Note that a boundary layer mesh can be obtained by computing thin layers near the boundary. Also, it has to be pointed out that the mesh of the outer surface is defined by the hexahedral elements. A similar method uses a harmonic field in order to precompute the fronts [150]. This method works for more general geometries and allows the user to choose the principal directions of the fronts.



(a) (b)

Figure 2.16: The receding front technique precompute fronts. (a) The fronts computed for a simple geometry and (b) a mesh obtained for a plane. (Pictures from [149].)

2.1.7 Indirect methods

Indirect methods transform an initial triangular/tetrahedral mesh into a quadrangular/hexahedral mesh. Triangular/tetrahedral mesh generators are fast, robust and fully automatic. The indirect approach has the advantage to work on any geometry and to allow making abstraction of the geometry definition. The transformations from triangles/tetrahedra to quadrangles/hexahedra can be done in two ways. The first is to split a triangle into 3 quadrangles or a tetrahedron into 4 hexahedra as shown in Figure 2.17. This technique is very simple to implement but produces bad-quality meshes so it is not actually used in this form. A more advanced method that works in 2D has been proposed in [151]. The algorithm classifies the edges (as either a boundary edge, a corner edge, etc). Then, some of the edges are split in two and triangles are split by applying splitting templates according to the classification of the edges.

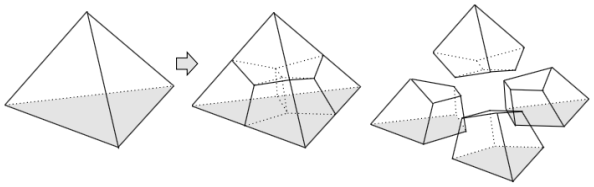


Figure 2.17: Element splitting. (Picture from [18].)

The other way to convert a simplicial mesh into a quadrangular/hexahedral mesh is by combinations. A quadrangle can be created by combining two adjacent triangles and a hexahedron can be created by combining usually 5 to 7 tetrahedra. In 2D, a first method was proposed in [152]. The method consists in combining pairs of triangles. The order in which they are combined influences the number of quadrangles that can be created and some heuristics have been presented in order to maximize the number of combinations. Reference [153] proposed additional topological operations (splitting and swapping) in order to improve the quality of the elements. The algorithm of [152] is improved in [154] by including local triangle splitting and by taking an advancing front approach. The front defines the limit between the already created quadrangles and the remaining triangles. This makes it possible to ensure that an all-quad mesh is created if the number of edges on the boundary is even. Another method is Q-Morph [155]. It improved boundary alignment and orthogonality by including edge swapping and by locally smoothing the mesh after a quadrangle is created. Also, quadrangles can be created by combining more than two triangles. Reference [156] proposes a combination technique that is able to respect a generalized metric map. The method does not take

an advancing front approach but achieves all-quadrangular meshing by splitting the elements of the resulting mixed mesh. Another approach that does not rely on an advancing fronts is proposed in [157], which takes advantage of the fact that combining pairs of triangles is equivalent to matching pairs of vertices in the dual graph of the mesh. An algorithm inspired from a well-known graph theory problem, *perfect matching*, is presented. First, a binary tree is extracted from the dual graph. Then, the matching is computed starting from the leaves and by adding *Steiner points* if necessary. This algorithm produces all-quad meshes but with a lot of irregular nodes. Based on this algorithm, reference [158] proposes a high-quality all-quadrangular mesh generator. To this end, the irregular nodes are converted to regular nodes by topological cleanups. In addition to common topological cleanups, an algorithm from [159] for remeshing patches of several elements is used.

In 3D, there is currently no method for creating all-hexahedral meshes on complex geometries. All the following techniques thus create hex-dominant meshes. A technique for searching the possible combinations of tetrahedra that form a hexahedron and combinations that form a prism is presented in [160]. The extension of Q-Morph in 3D has been presented in [161]. As for direct advancing front techniques, the method constructs conformal hex-dominant meshes in general since there remain tetrahedra that cannot be combined or transformed into valid hexahedra at meeting of fronts. Another approach which relies on a better node location of the initial tetrahedral mesh is presented in [162]. To do so, a *packing* of cubes is first computed, then the tetrahedral mesh is generated using an advancing front method. Note that this method does not allow respecting a prescribed size field. Another hex-dominant technique is presented in [163], where nodes are inserted from the boundary using a direction field and a size field, such that they are already good candidates for creating hexahedra. Then the tetrahedralization is computed and a greedy recombination algorithm is applied. At the end, a conformal hex-dominant mesh is obtained. Hexahedra that are near the boundaries are orthogonal and well-aligned. However, contrary to [161] non-hexahedral elements can remain at the boundaries of the geometry. Figure 2.18 shows a resulting mesh on a mechanical geometry.

2.2 Curved finite element validity and quality

An important characteristic of indirect methods which strongly influences the quality of the resulting mesh is the way the simplex to quadrangles/hexahedra transformations are applied. Indeed, the problem of combining the triangles/tetrahedra can be seen as a sequence of decisions to be made. Where to begin the combination process? Which triangles/tetrahedra to combine together and in which order? Answers to those questions depend on what is crucial for the simulation. For instance, alignment and orthogonality of ele-

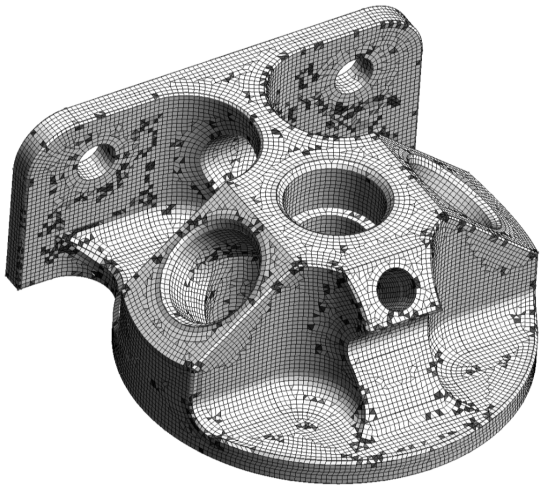


Figure 2.18: Mesh obtained by an indirect method. The mesh is conformal hex-dominant. Non-hexahedral elements are in grey while hexahedra are in white. (Picture from [163].)

ments can be highly desired near the boundaries of the geometry. In this case, it is better to combine first the triangles/tetrahedra that touch the boundaries. The (geometrical) validity and quality of the created quadrangles/hexahedra is certainly one of the most important characteristics that must lead the choices. Moreover, as it has been seen in the introduction, there is a real demand for the generation of curved high-order meshes and thus for methods to assess validity and quality of such meshes.

Concerning topological quality, it is known that a node valence 4 is ideal for quadrangular meshes. For hexahedra, it has been shown in [164] that node valence is less meaningful and that edge valence is a better indicator of topological quality. In the following, we review the state of the art for determining the geometrical validity of an element and measuring its quality (more details can be found in [75, 76, 165]). Most of the underlying techniques make use of the mapping between a reference element (defined *e.g.* in the unit square or the unit cube) and the actual (physical) mesh element defined in the computational domain. This mapping is used in FEM to recast all the spatial integrations that must be performed in the computational domain (*e.g.* coordinates (x, y)) as integrals in a (unique) reference coordinate system (*e.g.* coordinates (ξ, η), see Figure 2.19).

2.2.1 Methods for asserting the validity

It is known since the early age of FEM that the determinant of the Jacobian of the mapping between the reference and physical element has to be strictly

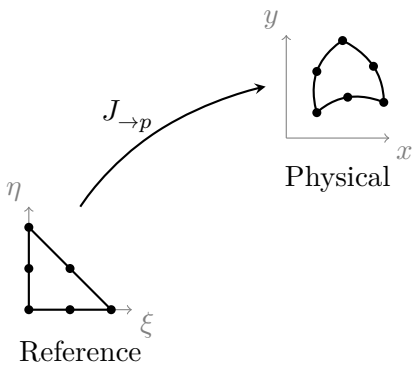


Figure 2.19: Mapping between the reference and the physical element.

positive in order for the mesh to be valid [166, 167]. For straight-sided simplicial elements, the Jacobian determinant is constant and for straight-sided quadrangles, the Jacobian determinant takes its minimum at one corner of the element, which implies that asserting the validity of those elements is trivial. This is not the case for hexahedra and more generally for curved (high-order) elements, for which the Jacobian determinant is a polynomial function of order at least two. The natural method for determining their validity is to compute the minimum of the Jacobian determinant, which remains straightforward for quadratic triangles [168] but not for the other elements.

Some validity conditions for the case of the quadratic serendipity (a.k.a. "8-node") quadrangle are given in [169] and an exact computation of the minimum of the Jacobian determinant for those elements is given in [170]. Although not being considered in those papers because it was not used in simulations at the time, the complete quadratic element, *i.e.* the "9-node" quadrangle can *a priori* be treated the same way. With additional geometrical tests, the computation can be accelerated in some cases [171]. Quadratic tetrahedra and linear hexahedra are studied in [168]. However, an exact computation is not achieved and only necessary conditions for the validity of those elements are provided. The case of the linear hexahedron is also studied in [172]. It is conjectured that if the Jacobian determinant is positive on the boundary, then it is also positive inside the element. However, even if it is straightforward to verify the positivity on the edges, no algorithm for verifying the positivity on the faces is given. Constructions relying on the positivity property of Bernstein-Bézier polynomials have been investigated in [173] for the case of quadratic tetrahedra. The Bézier coefficients of the Jacobian determinant are computed, which leads to a sufficient condition of validity if they are positive. The computed bound is however not sharp.

2.2.2 Geometric and Jacobian-based quality measures

Many quality measures have been proposed in the literature, particularly for linear simplicial elements. Several of these quality measures vanish when the element becomes invalid, which makes them also usable as a validity test. It is usually preferred to have normalized measures, *i.e.* measures that take their value between 0 and 1. However, measures that take negative values when the element is invalid are equivalently accepted. A distinction can be made between geometric quality measures and Jacobian-based quality measures.

Geometric quality measures

Used since the beginning of the FEM, geometric quality measures are constructed from geometric characteristics such as the area/volume of the element, the length of the edges or the radii of the inscribed and circumscribed circle/sphere. Reference [174] surveys many existing geometric quality measures for linear finite elements, while proposing additional quality measures. Reference [175] discusses the influence of the quality of the mesh on both the interpolation properties and the numerical solution of various PDEs and presents a comprehensive survey of geometric quality measures for linear simplicial elements. Geometric quality measures are not easily generalizable to curved elements. For quadratic tetrahedra, reference [173] proposes a measure constructed by approximating the volume and surface of the element by splitting it into 8 tetrahedra.

Let us also mention [176, 177], which studies the quality of second order triangles and tetrahedra. An algorithm is proposed for determining the admissible area in which the mid-side nodes can be located in such a way that the Jacobian determinant at the nodes of the element remains above a given value. A geometric quality measure is derived from this criterion; its generalization to higher orders is unclear.

Jacobian-based quality measures

The Jacobian matrix of the mapping between the reference or an "ideally-shaped" element and the physical element (see Figure 2.20) contains all the distortion information. It is natural to define quality measures from it; however it is only in the 2000s that such measures have been popularized. A great advantage of Jacobian-based measures is that their definition is independent of the type of element. They are also directly generalizable to curved elements. However, an important characteristic distinguishes the case of linear simplicial elements and other cases. The Jacobian matrix of linear simplicial elements is constant over the element, which implies that an element-wise quality measure can easily be constructed from it. For curved elements and even for linear quadrangular and hexahedral elements, the Jacobian matrix is not constant

over the element. Even if the same quality measure can be used, it will be only a pointwise measure. Thus, an additional step has to be applied in order to convert the pointwise measure into an element-wise measure.

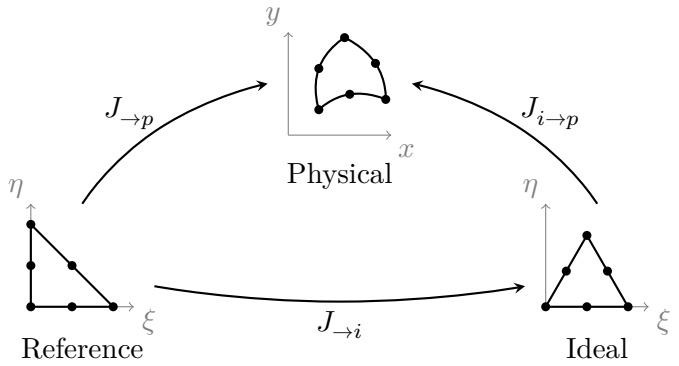


Figure 2.20: Mappings between the reference, the ideal and the physical element.

Linear elements A framework for algebraic quality measures is established in [178, 179]. This framework allows the construction, classification, and evaluation of such algebraic measures. From this framework, the definition of quality measures for simplicial elements is straightforward and several candidates are proposed. An element-wise measure Q is derived for quadrangular/hexahedral elements by computing the pointwise measure at all corner (4 in the case of a quadrangle and 8 in the case of an hexahedron) and by computing either their minimum or their harmonic or geometric average. This ensures that Q is 0 if and only if the pointwise quality measure vanishes at one corner and that it is equal to 1 if and only if the pointwise measure is 1 at all the corners.

For linear quadrangular/hexahedral elements the quality definition that prevails in the litterature is the *scaled Jacobian*. This measure appears to have been mentioned for the first time in [180] for quadrangles and in [181] for hexahedra. A geometrical definition can be found in [162]. The authors usually take the minimum of the scaled Jacobian computed at each corner as the element-wise quality measure.

Quadratic elements A quality measure for the quadratic serendipity (8-node) quadrangle, is proposed in [182]. The coordinates of the *face* node that would exist if the element was complete are derived in function of the coordinates of the 8 other nodes, and the parameters obtained in this derivation are used in the construction of a quality measure. Reference [183] considers the definition of algebraic quality measures for quadratic triangles. Similarly to

what had been proposed by the same author for linear quadrangles/hexahedra, it is proposed to compute the pointwise measure at every node of the element and to take the minimum, maximum or the p^{th} power-mean as the element-wise quality. In the case of the maximum, this global quality is compared to the actual maximum of the pointwise measure and it is shown that the former can be in some situations a poor approximation of the latter.

Arbitrary order elements Recent works have focused on defining a quality measure for curved simplicial finite elements of any order [184, 185]. The approach is to the consider an algebraic quality measure as proposed in [178], which constitutes the pointwise measure and to compute the L^2-norm as the element-wise quality measure. This measure however cannot be used as a validity test. It is mentioned in [186] that this technique, although developed only for simplicial elements, can be extended to non-simplicial elements. Note that using the same approach, algebraic quality measure can also be defined for non-planar curved triangles [187].

CHAPTER 3

Contributions

Despite the fact that tetrahedral mesh generation is significantly easier, we have seen in Section 2.1 that there are sufficient benefits of using all-hexahedral meshes in some applications to have a manifest demand for them [58]. However, we have also seen in Section 2.1.2 how this problem is highly constrained. Several successful methods (submapping, multisweeping, etc.) deal with these constraints by taking a "global approach". However, they are restrained to a certain class of geometries and they usually do not allow respecting a prescribed size field. A natural question is whether it is possible to design an algorithm for general geometries that has both property of global formulation and fine control of the element size.

Our first contribution, the *Blossom-Quad* algorithm, aims at finding a global approach which can respect a prescribed size field for the problem of quadrangulating general surfaces [39] (see Appendix A). The method takes an indirect approach; this way, the respect of the prescribed size field is trivial since the initial triangular mesh already respects it. The novelty of our first contribution is the formulation of the problem of combining pairs of triangles as a *minimum-cost perfect-matching problem* in the dual graph [1] of the triangular mesh. Minimum-cost perfect-matching is a well-known problem of graph theory and an efficient algorithm to solve it, called *Blossom*, has been originally proposed in [188, 189]. Our Blossom-Quad algorithm takes as input an initial mesh composed of an even number of connected triangles. Then, the potential combinations are listed and a cost is associated to each of them; this cost is function of the inverse of the quality measured on the quadrangle that would be created. The optimal solution of the combination problem is computed through the minimum-cost perfect-matching algorithm. In order

[1] We recall that the dual graph of a mesh is a graph whose vertices correspond to the elements of the mesh and whose edges correspond to the adjacency of two elements.

to ensure that a perfect match exists, "fake" combinations may occur at the boundary of the domain. Finally, a post-processing step is applied in order to deal with the possible fake combinations and to improve the quality of the final mesh [190, 191]. Blossom-Quad can be used with both CAD and STL models, provided for the latter that a reparameterization is performed (*e.g.* [192]). The algorithm is not perfect: (1) the fake combinations are not trivial to handle in order to have good quality elements at the boundaries of the geometry, and (2) even if multiple improvements have been proposed since 1965, the best known implementation of the minimum-cost perfect-matching algorithm in term of complexity runs in $O(n_{\triangle}^2)$ [193–195], where $n_{\triangle}$ is the number of triangles. The latter drawback can however be compensated by partitioning large meshes—at the cost of increasing the chances to obtain fake combinations. A typical timing of the Blossom-Quad algorithm is about 10 s for generating 50,000 quadrangles, out of which 3 s are spent for generating the original 100,000 triangles (on a Macbook Pro Retina, Mid 2012 @ 2.3GHz). Finally, we also show in the paper that the choice of the quality measure has a significant influence on the resulting mesh.

While this approach works very well it cannot be extended directly to the 3D problem of combining tetrahedra into hexahedra. Indeed, the graph theory matching problem is inherently only defined for "combining two elements together". A possible generalization is the reformulation of the matching problem but in this case, to the best of our knowledge, no corresponding, efficient algorithm exists. Moreover, it is not sufficient to find a "perfect combination" (*i.e.* a solution for which each tetrahedron is used for creating the hexahedra); the solution also has to be conformal, which adds other difficulties. Those conformity constraints can be nicely handled by constructing the graph of compatibilities between potential hexahedra or its complementary, the graph of incompatibilities between potential hexahedra. Searching a solution in those graphs corresponds to two well-known problems of graph theory: the *maximum clique* and *maximum independent set* problem respectively [196]. However, those two problems are NP-complete in general and can be solved in polynomial time only for particular graph structures.

Our second contribution aims at finding an alternative approach that can easily be extended to 3D [40] (see Appendix B). To this end, we still take an indirect approach and we formulate the combination problem as a *sequential decision-making problem*, which is a problem often come across in reinforcement learning, a domain of artificial intelligence [197]. The quadrangulation process is thus viewed as a sequence of combinations to perform. In contrast to the techniques reviewed in Section 2.1.7, which use pure heuristics to determine the combinations to add in the sequence, we employ a *look-ahead tree* [198]. This look-ahead tree allows, at a given state of the mesh (*i.e.* at an intermediate state between the initial triangular mesh and the final mesh),

to attach a "reward"—that is based not only on the quality of the quadrangle that would be created but also on the quality of the future quadrangles that can be further created in the neighborhood—to each potential combination. In other words, the look-ahead tree allows to "see in the future" and to anticipate, in a similar way a human would do it. Heuristics are used in order to obtain as much quadrangles as possible but patches of triangles that cannot be further combined usually remain. However, they contain an even number of triangles and the final mesh can easily be post-processed in order to be transformed into an all-quadrangular mesh (*e.g.* using [159]). We show that augmenting the depth of the tree, up to a certain number, increases the quality of the final mesh as well as the number of combinations. This method can be seen as a relaxation of Blossom-Quad, since it computes a near-optimal solution, while Blossom-Quad computes the optimal one. However, contrary to Blossom-Quad which has a complexity of $O(n_{\triangle}^2)$, this algorithm is linear and allows to achieve better quality elements at the boundaries. Moreover, it has the advantage to be more flexible: additional topological and geometrical operations can be applied during the combination process. Finally, the application of the look-ahead tree technique on the 3D problem is straightforward. A typical percentage of combination success of this algorithm is between 98% and 99%, in function of the depth of the look-ahead tree. The timing is typically 16 s for combining 100,000 triangles with a look-ahead tree of depth 3.

In both of these methods, the location of the nodes in the initial triangular mesh is not insignificant; they should ideally be placed such that the triangles are good candidates for the combinations. Traditional triangular mesh generators usually aim at producing triangles that are as equilateral as possible. For the combination problem, it is usually best if the triangles are closer to right-angled triangles. Several techniques can be used for this purpose [199–201].

Also, in both of these approaches, the functional to optimize is based on the validity and the quality of the created quadrangles. In practice however, there exists currently no method to robustly assert that a (linear) hexahedron is valid. Moreover, it is unclear if computing the minimum of the scaled Jacobian at the corner of linear hexahedra is sufficient to judge their quality. The next three papers deal with this validity issue, as well as proposing a more robust quality measure. The scope of these contributions is actually much larger, as they address the validity and quality of arbitrary order finite elements, which are crucial for the development of the new generation of high-order solvers.

In our third contribution, we design an efficient algorithm to accurately compute the minimum and the maximum of the Jacobian determinant (to which we refer as $J_{\min}$ and $J_{\max}$ respectively) of any common curvilinear fi-

nite elements except the pyramid [41] (see Appendix C). Since the Jacobian determinant is a polynomial function, we expand it in the Bézier basis that corresponds to the element. Bézier bases having both properties of positivity and boundedness, the minimum (resp. maximum) of the expansion coefficients constitutes a lower (resp. upper) bound of the Jacobian determinant, which is however not sharp in general. To sharpen those bounds we employ a method which consists in *subdividing* the expansion. This subdivision is done in a recursive and adaptive manner and has a quadratic rate of convergence [202]. In order to be able to characterize the sharpness of, *e.g.*, the lower bound, an upper bound of $J_{\min}$ is necessary. This upper bound is provided by actual values of the Jacobian determinant amongst the coefficients of the expansion. Experiments show that the extrema of the Jacobian determinant can be computed efficiently with any prescribed tolerance. This algorithm allows then to assert for certain that an element is valid or not by looking at the sign of $J_{\min}$. Moreover, the minimum and the maximum of the Jacobian determinant can be used to construct a distortion measure. In practice, the method can determine the validity of 1,000,000 linear hexahedra in about 2.5 s while the same number of third-order tetrahedra are processed in 9 s. Note that this work is based in part on our master thesis [203].

Our fourth contribution extends this method to the case of Bergot's curvilinear pyramids [42] (see Appendix D). For pyramids, the Jacobian determinant is a rational function and this extension is not trivial [204]. However, we show that a Bézier-like basis can be constructed with the same positivity and boundedness properties. This basis can be used for optimization but does not allow subdivision; we thus construct an enriched basis for which the algorithm of our third contribution can apply. Results show that the method is as efficient as for the other types of elements.

Our last contribution, which is still a draft paper, aims at efficiently computing the extrema of a (pointwise) Jacobian-based quality measure for any arbitrary-order pyramidal and non-pyramidal elements (see Appendix E). We consider the eigenvalues of the *metric tensor* (which correspond to the singular values of the Jacobian matrix) and we take the ratio between the minimum and the maximum of those eigenvalues as the quality measure. This quality measure has been shown to be effective for simulations [36]. Geometrically, this is a measure of the pointwise anisotropy within the curved element. Since the metric tensor is a real symmetric matrix, in the 3D case, a trigonometric expression of the eigenvalues can be established by an affine transformation [205]. Despite its relative complexity, the method takes typically only about 20 to 30 times longer to fully analyze the quality of linear hexahedral or third-order tetrahedral meshes than the corresponding analysis of their validity. This makes the computational cost of the algorithm comparable to the mesh generation time.

The corresponding algorithms are integrated to the stable release of the software Gmsh [57] which can be downloaded on the following website: `www.gmsh.info`. Since Gmsh is an open-source software, all the algorithms can be found in its source code:

- The Blossom-Quad algorithm is currently the default algorithm for creating quadrangular meshes with the indirect approach and the code can be found in `gmsh/Mesh/meshGFaceOptimize.{h/cpp}`.
- The algorithm of our second contribution is still experimental and can only be used through the C++ API: The code is located in `gmsh/Mesh/meshGFaceRecombine.{h/cpp}`.
- The algorithm for detecting invalid curvilinear elements as well as the algorithm for computing their quality can be used through the plugin `AnalyseCurvedMesh`. The implementation is scattered in multiple files that are mainly located in directory `gmsh/Numeric` but a good starting point is to look at `gmsh/Plugin/AnalyseCurvedMesh.{h/cpp}`.

CHAPTER 4

Conclusions

The objective of this thesis was twofold: (1) develop efficient quadrangulation techniques for finite element applications, in view of their eventual extension to the very hard problem of automatic all-hexahedral mesh generation; and (2) design accurate procedures for assessing the validity and the quality of any type of finite element.

As was reviewed in Chapter 2, numerous techniques have been proposed over the years to generate quadrangular/hexahedral meshes. We have focused on indirect methods, which inherit their robustness and ability to respect prescribed size fields from the underlying triangulation/tetrahedralization, and have investigated two research directions in this setting. The first one consists in formulating the combination problem as a *minimum-cost perfect-matching problem*. The main advantage is that this allows computing an optimal solution for which all the triangles are combined. In order for this to be possible, the only condition is that the number of triangles in the initial mesh is even. However, some "fake" combinations at the boundaries may occur and have to be handled as a post-processing step. The second approach consists in using a *look-ahead tree technique* to determine the triangles to combine. This technique allows to obtain a near-optimal solution; in this sense, it is between approaches based on pure heuristics and the optimal approach. By augmenting the depth of the tree, the quality of the final mesh increases, without increasing the complexity of the algorithm that remains linear. We believe that current indirect quadrangulation methods have reached a plateau due to the difficulty to further improve the heuristics and that this look-ahead tree technique, which adds "computer intelligence", is a very promising way forward, especially as it is directly applicable in 3D.

For the second problem that we have investigated, we have shown that, using the properties of Bézier expansions, it is possible to efficiently compute the

extrema values of Jacobian-based quantities defined on finite elements of any order and type (including pyramidal elements). In particular, the minimum of the Jacobian determinant can be efficiently computed with any prescribed tolerance, which allows asserting the validity of curvilinear elements. The same technique is the basis of the proposed method for the calculation of a quality measure that quantifies the pointwise anisotropy of the elements. We believe that this strategy constitutes a fundamental building block for the analysis and optimization of finite element meshes.

Perspectives

Many perspectives for future works exist.

For quadrangular mesh generation:

1. *Extension of the quadrangulation techniques to 3D*: The Blossom-Quad algorithm gives nice results, however, as mentioned before, its generalization to 3D is problematic and the generation of high-quality quadrangles near the boundaries is difficult. The application of the look-ahead tree technique to 3D is on the contrary trivial, but several questions arise, in particular due to the fact that the branching factor of the tree will be much larger for combining hexahedra than for combining quadrangles (see 3. below).

2. *Study provability of the validity and quality of indirect all-quadrangular (and all-hexahedral) mesh generation*: As explained in Section 2.2, a valid and good quality mesh is essential for finite element simulations. When creating a quad-dominant mesh a simple solution is to only combine triangles that lead to valid/good-quality quadrangles. When an all-quadrangular mesh is needed, no trivial solution exists.

3. *Investigation of an advanced look-ahead tree technique*: We have shown that increasing the depth of the tree leads to an improvement of the resulting mesh. However, expanding the tree uniformly is expensive, since the computation time increases proportionally to h^α, with h the depth and α the branching factor. While the branching factor is reasonably small for the 2D problem (< 2), it is expected to be much larger for the 3D problem (> 10). Therefore, it is crucial to investigate more advanced look-ahead tree techniques such as the *Monte Carlo* tree search [206].

4. *Additional types of actions*: While the use of a triangular mesh generator that produces close to right-angled triangles, such as [200], is very useful, especially near the boundaries, the node location is however not optimal and many bad-quality or even invalid quadrangles are created. Thus, in

addition to the combinations, it may be useful to add other actions such as node relocation, edge swaping, or collapse.

For the validation of meshes:

1. *Extension of the method to other quality measures*: The ability to compute the extrema of the Jacobian determinant and the minimum of a Jacobian-based quality measure suggests that it should be possible to extend the technique to other Jacobian-based quality measures. In particular, it would be very helpful to be able to compute the minimum of the scaled Jacobian, which is the reference quality measure used in the hexahedral mesh generation community.

2. *Study of the proposed element-wise quality measures*: There are several studies on the influence of the geometric quality of linear elements on the FEM solution. However, the influence of the quality of curved elements on FEM solutions is still largely unknown. The advent of good quality measures should help greatly in such studies.

3. *Use of the results for optimizing curvilinear meshes*: It has already be shown that Bézier expansions can also be used in an optimization algorithm to make valid (*to untangle*) a curved mesh [36]. One question that arises is whether it is possible to do the same for optimizing the quality of the mesh.

Bibliography

[1] E. R. Arantes Oliveira. Theoretical foundations of the finite element method. *International Journal of Solids and Structures*, 4(10):929–952, 1968.

[2] C. V. Girijavallabhan and L. C. Reese. Finite-element method for problems in soil mechanics. *Journal of Soil Mechanics & Foundations Div*, 94(2):473–496, 1968.

[3] R. S. Sandhu and E. L. Wilson. Finite-element analysis of seepage in elastic media. *Journal of the Engineering Mechanics Division*, 95(3): 641–652, 1969.

[4] C. A. Brebbia and J. J. Connor. *Fundamentals of finite element techniques for structural engineers*. Butterworth Press, 1973.

[5] G. Strang. Piecewise polynomials and the finite element method. *Bulletin of the American Mathematical Society*, 79(6):1128–1137, 1973.

[6] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. "Consider a spherical cow" – Conservation of geometry in analysis: Implications for computational methods in engineering. IMA Hot Topics Workshop: Compatible Spatial Discretizations for Partial Differential Equations, 2004.

[7] D. Roylance. Finite element analysis. MIT Press, Cambridge, 2001.

[8] I. Babuška, W. D. Henshaw, J. E. Oliger, J. E. Flaherty, J. E. Hopcroft, and T. Tezduyar, editors. *Modeling, mesh generation, and adaptive numerical methods for partial differential equations*. Springer Science & Business Media, 1995.

[9] K. Haghighi and E. Kang. A knowledge-based approach to the adaptive finite element analysis. In *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, pages 267–276. Springer, 1995.

[10] B.-Y. Shih and H. Sakurai. Automated hexahedral mesh generation by swept volume decomposition and recomposition. In *Proceedings of*

the 5th International Meshing Roundtable, volume 96, pages 273–280. Citeseer, 1996.

[11] T. Blacker. Automated conformal hexahedral meshing constraints, challenges and opportunities. *Engineering with Computers*, 17(3):201–210, 2001.

[12] J. F. Shepherd and C. R. Johnson. Hexahedral mesh generation constraints. *Engineering with Computers*, 24(3):195–213, 2008.

[13] S. J. Owen, B. W. Clark, D. J. Melander, M. Brewer, J. F. Shepherd, K. Merkley, C. Ernst, and R. Morris. An immersive topology environment for meshing. In *Proceedings of the 16th International Meshing Roundtable*, pages 553–577. Springer, 2008.

[14] T. J. Baker. Mesh generation: Art or science? *Progress in Aerospace Sciences*, 41(1):29–63, 2005.

[15] J. Chen, D. Zhao, Y. Zheng, Z. Huang, and J. Zheng. Fine-grained parallel algorithm for unstructured surface mesh generation. In *Proceedings of the 22nd International Meshing Roundtable*, pages 559–578. Springer, 2014.

[16] S. A. Canann. *Plastering and optismoothing: New approaches to automated, 3D hexahedral mesh generation and mesh smoothing.* PhD thesis, Brigham Young University, 1991.

[17] D. R. White. *Automatic, quadrilateral and hexahedral meshing of pseudo-cartesian geometries using virtual subdivision.* PhD thesis, Brigham Young University, 1996.

[18] S. J. Owen. *Non-simplical unstructured mesh generation.* PhD thesis, Carnegie Mellon University, 1999.

[19] O. U. Baran. *Control methodologies in unstructured hexahedral grid generation.* PhD thesis, Vrije Universiteit Brussel, 2005.

[20] K. Kovalev. *Unstructured hexahedral non-conformal mesh generation.* PhD thesis, Vrije Universiteit Brussel, 2005.

[21] J. F. Shepherd. *Topologic and geometric constraint-based hexahedral mesh generation.* PhD thesis, The University of Utah, 2007.

[22] X. Roca. *Paving the path towards automatic hexahedral mesh generation.* PhD thesis, Universitat Politècnica de Catalunya, 2009.

[23] E. Ruiz-Gironés. *Automatic hexahedral meshing algorithms: from structured to unstructured meshes.* PhD thesis, Universitat Politècnica de Catalunya, 2011.

[24] T. Carrier Baudouin. *Hexahedral-dominant mesh generation*. PhD thesis, Université catholique de Louvain, 2013.

[25] I. Babuška, B. A. Szabo, and I. N. Katz. The p-version of the finite element method. *SIAM Journal on Numerical Analysis*, 18(3):515–545, 1981.

[26] I. Babuška and B. Q. Guo. The h-p version of the finite element method for domains with curved boundaries. *SIAM Journal on Numerical Analysis*, 25(4):837–861, 1988.

[27] R. H. MacNeal. *Finite Elements*. CRC Press, 1993.

[28] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, et al. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

[29] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[30] R. M. Kirby, S. J. Sherwin, and B. Cockburn. To CG or to HDG: a comparative study. *Journal of Scientific Computing*, 51(1):183–212, 2012.

[31] P. E. J. Vos, S. J. Sherwin, and R. M. Kirby. From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low-and high-order discretisations. *Journal of Computational Physics*, 229(13):5161–5181, 2010.

[32] G. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

[33] P. M. Knupp. Winslow smoothing on two-dimensional unstructured meshes. *Engineering with Computers*, 15(3):263–268, 1999.

[34] Z. Q. Xie, R. Sevilla, O. Hassan, and K. Morgan. The generation of arbitrary order curved meshes for 3D finite element analysis. *Computational Mechanics*, 51(3):361–374, 2013.

[35] S. J. Sherwin and J. Peiró. Mesh generation in curvilinear domains using high-order elements. *International Journal for Numerical Methods in Engineering*, 53(1):207–223, 2002.

[36] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26, 2013.

[37] A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate. Optimization of a regularized distortion measure to generate curved high-order unstructured tetrahedral meshes. *International Journal for Numerical Methods in Engineering*, 2015.

[38] C. Geuzaine, A. Johnen, J. Lambrechts, J.-F. Remacle, and T. Toulorge. The generation of valid curvilinear meshes. In *IDIHOM: Industrialization of High-Order Methods-A Top-Down Approach*, pages 15–39. Springer, 2015.

[39] J.-F. Remacle, J. Lambrechts, B. Seny, E. Marchandise, A. Johnen, and C. Geuzaine. Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. *International Journal for Numerical Methods in Engineering*, 89(9):1102–1119, 2012.

[40] A. Johnen, D. Ernst, and C. Geuzaine. Sequential decision-making approach for quadrangular mesh generation. *Engineering with Computers*, 31(4):729–735, 2015.

[41] A. Johnen, J.-F. Remacle, and C. Geuzaine. Geometrical validity of curvilinear finite elements. *Journal of Computational Physics*, 233:359–372, 2013.

[42] A. Johnen and C. Geuzaine. Geometrical validity of curvilinear pyramidal finite elements. *Journal of Computational Physics*, 299:124–129, 2015.

[43] L. F. Richardson. *Weather prediction by numerical process*. Cambridge University Press, 1922.

[44] F. Ledoux and J. Shepherd. Topological and geometrical properties of hexahedral meshes. *Engineering with Computers*, 26(4):419–432, 2010.

[45] R. Löhner and P. Parikh. Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8(10):1135–1149, 1988.

[46] M. S. Shephard and M. K. Georges. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering*, 32(4):709–749, 1991.

[47] P. L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering*, 92(3):269–288, 1991.

[48] S. Rebay. Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm. *Journal of Computational Physics*, 106(1):125–138, 1993.

[49] P. L. George and É. Seveno. The advancing-front mesh generation method revisited. *International Journal for Numerical Methods in Engineering*, 37(21):3605–3619, 1994.

[50] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.

[51] R. Löhner. Progress in grid generation via the advancing front technique. *Engineering with Computers*, 12(3-4):186–210, 1996.

[52] J. R. Shewchuk. *Delaunay refinement mesh generation*. PhD thesis, Carnegie Mellon University, 1997.

[53] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry Towards Geometric Engineering*, pages 203–222. Springer, 1996.

[54] J.-D. Boissonnat, O. Devillers, M. Teillaud, and M. Yvinec. Triangulations in CGAL. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 11–18. ACM, 2000.

[55] H. Si. *TetGen: A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator*, 2004.

[56] P. L. George, F. Hecht, and E. Saltel. TetMesh-Ghs3d, mesh generator for tetrahedral element, 2004.

[57] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[58] J. Sarrate, E. Ruiz-Gironés, and X. Roca. Unstructured and semi-structured hexahedral mesh generation methods. *Computational Technology Reviews*, 10:35–64, 2014.

[59] A. O. Cifuentes and A. Kalbag. A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. *Finite Elements in Analysis and Design*, 12(3):313–318, 1992.

[60] V. I. Weingarten. The controversy over hex or tet meshing. *Machine Design*, 66(8):74–76, 1994.

[61] S. E. Benzley, E. Perry, K. Merkley, B. Clark, and G. Sjaardama. A comparison of all-hexagonal and all-tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings of the 4th International Meshing Roundtable*, 1995.

[62] A. Huerta, A. Angeloski, X. Roca, and J. Peraire. Efficiency of high-order elements for continuous and discontinuous Galerkin methods. *International Journal for Numerical Methods in Engineering*, 96(9):529–560, 2013.

[63] R. Biswas and R. C. Strawn. Tetrahedral and hexahedral mesh adaptation for CFD problems. *Applied Numerical Mathematics*, 26(1):135–151, 1998.

[64] D. Komatitsch, S. Tsuboi, and J. Tromp. The spectral-element method in seismology. *Geophysical Monograph Series*, 157:205–227, 2005.

[65] M. L. Staten, R. A. Kerr, S. J. Owen, T. D. Blacker, M. Stupazzini, and K. Shimada. Unconstrained plastering—Hexahedral mesh generation via advancing-front geometry decomposition. *International Journal for Numerical Methods in Engineering*, 81(2):135–171, 2010.

[66] J.-F. Remacle, R. Gandham, and T. Warburton. Gpu accelerated spectral finite elements on all-hex meshes. *arXiv preprint arXiv:1506.05996*, 2015.

[67] R. Schneiders, R. Schindler, and F. Weiler. Octree-based generation of hexahedral element meshes. In *Proceedings of the 5th International Meshing Roundtable*. Citeseer, 1996.

[68] Y. Ito, A. M. Shih, and B. K. Soni. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. *International Journal for Numerical Methods in Engineering*, 77(13): 1809–1833, 2009.

[69] P. Murdoch, S. Benzley, T. Blacker, and S. A. Mitchell. The spatial twist continuum: A connectivity based method for representing all-hexahedral finite element meshes. *Finite Elements in Analysis and Design*, 28(2): 137–149, 1997.

[70] S. A. Mitchell. A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volume. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, pages 465–476. Springer, 1996.

[71] R. Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering with Computers*, 12(3-4):168–177, 1996.

[72] T. Suzuki, S. Takahashi, and J. Shepherd. An interior surface generation method for all-hexahedral meshing. In *Proceedings of the 14th International Meshing Roundtable*, pages 377–398. Springer, 2005.

[73] S. Yamakawa and K. Shimada. Hexhoop: Modular templates for converting a hex-dominant mesh to an all-hex mesh. *Engineering with Computers*, 18(3):211–228, 2002.

[74] S. Yamakawa and K. Shimada. 88-element solution to Schneiders' pyramid hex-meshing problem. *International Journal for Numerical Methods in Biomedical Engineering*, 26(12):1700–1712, 2010.

[75] J. F. Thompson, B. K. Soni, and N. P. Weatherill. *Handbook of grid generation*. CRC press, 1998.

[76] P. J. Frey and P.-L. George. *Mesh generation: application to finite elements*. ISTE, 2008.

[77] W. J. Gordon and C. A. Hall. Construction of curvilinear co-ordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering*, 7(4):461–477, 1973.

[78] R. Haber, M. S. Shephard, J. F. Abel, R. H. Gallagher, and D. P. Greenberg. A general two-dimensional, graphical finite element preprocessor utilizing discrete transfinite mappings. *International Journal for Numerical Methods in Engineering*, 17(7):1015–1044, 1981.

[79] W. A. Cook and W. R. Oakes. Mapping methods for generating three-dimensional meshes. *Computers in Mechanical Engineering*, 8:67–72, 1982.

[80] T. Shih, R. T. Bailey, H. L. Nguyen, R. J. Roelke, et al. Algebraic grid generation for complex geometries. *International Journal for Numerical Methods in Fluids*, 13(1):1–31, 1991.

[81] C. C. L. Sells. Plane subcritical flow past a lifting aerofoil. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 308(1494):377–401, 1969.

[82] J. South and A. Jameson. Relaxation solutions for inviscid axisymmetric transonic flow over blunt or pointed objects. In *AIAA Computational Fluid Dynamics Conference*, 1973.

[83] D. C. Ives. Conformal grid generation. *Applied Mathematics and Computation*, 10:107–135, 1982.

[84] A. M. Winslow. Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh. *Journal of Computational Physics*, 1(2): 149–172, 1966.

[85] J. F. Thompson, F. C. Thames, and C. W. Mastin. Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies. *Journal of Computational Physics*, 15(3):299–319, 1974.

[86] M. B. Stephenson and T. D. Blacker. Using conjoint meshing primitives to generate quadrilateral and hexahedral elements in irregular regions. Technical report, Sandia National Labs., Albuquerque, NM (USA), 1989.

[87] T. S. Li, R. M. McKeag, and C. G. Armstrong. Hexahedral meshing using midpoint subdivision and integer programming. *Computer Methods in Applied Mechanics and Engineering*, 124(1):171–193, 1995.

[88] D. R. White and T. J. Tautges. Automatic scheme selection for toolkit hex meshing. *International Journal for Numerical Methods in Engineering*, 49(1-2):127–144, 2000.

[89] X. Roca, J. Sarrate, and A. Huerta. Mesh projection between parametric surfaces. *Communications in Numerical Methods in Engineering*, 22(6): 591–603, 2006.

[90] P. M. Knupp. Applications of mesh smoothing: copy, morph, and sweep on unstructured quadrilateral meshes. *International Journal for Numerical Methods in Engineering*, 45(1):37–45, 1999.

[91] L. Mingwu, S. E. Benzley, G. Sjaardema, and T. Tautges. A multiple source and target sweeping method for generating all-hexahedral finite element meshes. In *Proceedings of the 5th International Meshing Roundtable*, volume 96, pages 217–225. Citeseer, 1996.

[92] P. M. Knupp. Next-Generation Sweep Tool: A method for generating all-hex meshes on two-and-one-half dimensional geometries. In *Proceedings of the 7th International Meshing Roundtable*, pages 505–513, 1998.

[93] M. L. Staten, S. A. Canann, and S. J. Owen. BMSweep: Locating interior nodes during sweeping. *Engineering with Computers*, 15(3): 212–218, 1999.

[94] X. Roca and J. Sarrate. An automatic and general least-squares projection procedure for sweep meshing. *Engineering with Computers*, 26(4): 391–406, 2010.

[95] X. Roca and J. Sarrate. Least-squares approximation of affine mappings for sweep mesh generation: functional analysis and applications. *Engineering with Computers*, 29(1):1–15, 2013.

[96] T. Blacker. The cooper tool. In *Proceedings of the 5th International Meshing Roundtable*. Citeseer, 1996.

[97] D. R. White, S. Saigal, and S. J. Owen. CCSweep: Automatic decomposition of multi-sweep volumes. *Engineering with Computers*, 20(3): 222–236, 2004.

[98] M. A. Scott, M. N. Earp, S. E. Benzley, and M. B. Stephenson. Adaptive sweeping techniques. In *Proceedings of the 14th International Meshing Roundtable*, pages 417–432. Springer, 2005.

[99] D. R. White, L. Mingwu, S. E. Benzley, and G. D. Sjaardema. Automated hexahedral mesh generation by virtual decomposition. In *Proceedings of the 4th International Meshing Roundtable*, pages 165–176. Citeseer, 1995.

[100] M. Whiteley, D. White, S. Benzley, and T. Blacker. Two and three-quarter dimensional meshing facilitators. *Engineering with Computers*, 12(3-4):144–154, 1996.

[101] E. Ruiz-Gironés and J. Sarrate. Generation of structured hexahedral meshes in volumes with holes. *Finite Elements in Analysis and Design*, 46(10):792–804, 2010.

[102] E. Ruiz-Gironés and J. Sarrate. Generation of structured meshes in multiply connected surfaces using submapping. *Advances in Engineering Software*, 41(2):379–387, 2010.

[103] R. Chen and P. Xi. A digraph-based hexahedral meshing method for coupled quasi-polycubes. *Computer Methods in Applied Mechanics and Engineering*, 268:18–39, 2014.

[104] J. Shepherd, S. Benzley, and S. A. Mitchell. Interval assignment for volumes with holes. *International Journal for Numerical Methods in Engineering*, 49(1-2):277–288, 2000.

[105] M. Lai, S. Benzley, and D. White. Automated hexahedral mesh generation by generalized multiple source to multiple target sweeping. *International Journal for Numerical Methods in Engineering*, 49(1-2):261–275, 2000.

[106] J. F. Shepherd, S. A. Mitchell, P. Knupp, and D. R. White. Methods for multisweep automation. In *Proceedings of the 14th International Meshing Roundtable*, 2000.

[107] E. Ruiz-Gironés, X. Roca, and J. Sarrate. Using a computational domain and a three-stage node location procedure for multi-sweeping algorithms. *Advances in Engineering Software*, 42(9):700–713, 2011.

[108] M. A. Scott, S. E. Benzley, and S. J. Owen. Improved many-to-one sweeping. *International Journal for Numerical Methods in Engineering*, 65(3):332–348, 2006.

[109] K. Miyoshi and T. D. Blacker. Hexahedral mesh generation using multi-axis cooper algorithm. In *Proceedings of the 9th International Meshing Roundtable*. Citeseer, 2000.

[110] P. Sampl. Medial axis construction in three dimensions and its application to mesh generation. *Engineering with Computers*, 17(3):234–248, 2001.

[111] D. J. Sheehy, C. G. Armstrong, and D. J. Robinson. Computing the medial surface of a solid from a domain Delaunay triangulation. In *Proceedings of the third ACM symposium on Solid modeling and applications*, pages 201–212. ACM, 1995.

[112] G. M. Turkiyyah, D. W. Storti, M. Ganter, H. Chen, and M. Vimawala. An accelerated triangulation method for computing the skeletons of free-form solid models. *Computer-Aided Design*, 29(1):5–19, 1997.

[113] T. K. H. Tam and C. G. Armstrong. 2D finite element mesh generation by medial axis subdivision. *Advances in Engineering Software and Workstations*, 13(5):313–324, 1991.

[114] M. A. Price, C. G. Armstrong, and M. A. Sabin. Hexahedral mesh generation by medial surface subdivision: Part I. Solids with convex edges. *International Journal for Numerical Methods in Engineering*, 38 (19):3335–3359, 1995.

[115] M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part II. Solids with flat and concave edges. *International Journal for Numerical Methods in Engineering*, 40(1):111–136, 1997.

[116] A. Sheffer, M. Etzion, A. Rappoport, and M. Bercovier. Hexahedral mesh generation using the embedded Voronoi graph. *Engineering with Computers*, 15(3):248–262, 1999.

[117] X.-J. Luo, M. S. Shephard, L.-Z. Yin, R. M. O'Bara, R. Nastasi, and M. W. Beall. Construction of near optimal meshes for 3D curved domains with thin sections and singularities for p-version method. *Engineering with Computers*, 26(3):215–229, 2010.

[118] T. T. Robinson, C. G. Armstrong, and R. Fairey. Automated mixed dimensional modelling from 2D and 3D CAD models. *Finite Elements in Analysis and Design*, 47(2):151–165, 2011.

[119] J. E. Makem, C. G. Armstrong, and T. T. Robinson. Automatic decomposition and efficient semi-structured meshing of complex solids. *Engineering with Computers*, 30(3):345–361, 2014.

[120] W. R. Quadros. LayTracks3D: Mesh generator for general assembly models using medial axis transform. In *Research notes of the 22nd International Meshing Roundtable*, 2013.

[121] J. H.-C. Lu, I. Song, W. R. Quadros, and K. Shimada. Geometric reasoning in sketch-based volumetric decomposition framework for hexahedral meshing. *Engineering with Computers*, 30(2):237–252, 2014.

[122] F. Kälberer, M. Nieser, and K. Polthier. QuadCover - Surface parameterization using branched coverings. *Computer Graphics Forum*, 26(3): 375–384, 2007.

[123] J. Palacios and E. Zhang. Rotational symmetry field design on surfaces. *ACM Transactions on Graphics (TOG)*, 26(3):55, 2007.

[124] D. Bommes, H. Zimmer, and L. Kobbelt. Mixed-integer quadrangulation. *ACM Transactions on Graphics (TOG)*, 28(3):77, 2009.

[125] N. Ray, B. Vallet, L. Alonso, and B. Lévy. Geometry-aware direction field processing. *ACM Transactions on Graphics (TOG)*, 29(1):1, 2009.

[126] H. J. Fogg, C. G. Armstrong, and T. T. Robinson. Multi-block decomposition using cross-fields. In *VI International Conference on Adaptive Modeling and Simulation*, pages 254–267, 2013.

[127] N. Kowalski, F. Ledoux, and P. Frey. Automatic domain partitioning for quadrilateral meshing with line constraints. *Engineering with Computers*, pages 1–17, 2014.

[128] G. Bunin. A continuum theory for unstructured mesh generation in two dimensions. *Computer Aided Geometric Design*, 25(1):14–40, 2008.

[129] M. Nieser, U. Reitebuch, and K. Polthier. CubeCover – Parameterization of 3D volumes. *Computer Graphics Forum*, 30(5):1397–1406, 2011.

[130] Y. Li, Y. Liu, W. Xu, W. Wang, and B. Guo. All-hex meshing using singularity-restricted field. *ACM Transactions on Graphics (TOG)*, 31 (6):177, 2012.

[131] J. Huang, Y. Tong, H. Wei, and H. Bao. Boundary aligned smooth 3D cross-frame field. *ACM Transactions on Graphics (TOG)*, 30(6):143, 2011.

[132] J. Huang, T. Jiang, Y. Wang, Y. Tong, and H. Bao. Automatic frame field guided hexahedral mesh generation. Technical report, State Key Lab of CAD & CG, College of Computer Science at Zhejiang University, 2012.

[133] N. Kowalski, F. Ledoux, and P. Frey. Smoothness driven frame field generation for hexahedral meshing. *Computer-Aided Design*, In press.

[134] R. Schneiders and R. Bünten. Automatic generation of hexahedral finite element meshes. *Computer Aided Geometric Design*, 12(7):693–707, 1995.

[135] J. Qian and Y. Zhang. Automatic unstructured all-hexahedral mesh generation from B-Reps for non-manifold CAD assemblies. *Engineering with Computers*, 28(4):345–359, 2012.

[136] Y. Zhang, C. Bajaj, and B.-S. Sohn. 3D finite element meshing from imaging data. *Computer Methods in Applied Mechanics and Engineering*, 194(48):5083–5106, 2005.

[137] Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering*, 195(9):942–960, 2006.

[138] Y. Zhang, T. J. R. Hughes, and C. L. Bajaj. An automatic 3D mesh generation method for domains with multiple materials. *Computer Methods in Applied Mechanics and Engineering*, 199(5):405–415, 2010.

[139] J. Qian, Y. Zhang, W. Wang, A. C. Lewis, M. A. Qidwai, and A. B. Geltmacher. Quality improvement of non-manifold hexahedral meshes for critical feature determination of microstructure materials. *International Journal for Numerical Methods in Engineering*, 82(11):1406–1423, 2010.

[140] S. J. Owen and J. F. Shepherd. Embedding features in a cartesian grid. In *Proceedings of the 18th International Meshing Roundtable*, pages 117–138. Springer, 2009.

[141] T. D. Blacker and M. B. Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32(4):811–847, 1991.

[142] T. D. Blacker, M. B. Stephenson, and S. Canann. Analysis automation with paving: A new quadrilateral meshing technique. *Advances in Engineering Software and Workstations*, 13(5):332–337, 1991.

[143] D. R. White and P. Kinney. Redesign of the paving algorithm: robustness enhancements through element by element meshing. In *Proceedings of the 6th International Meshing Roundtable*, pages 323–335, 1997.

[144] R. J. Cass, S. E. Benzley, R. J. Meyers, and T. D. Blacker. Generalized 3-D paving: An automated quadrilateral surface mesh generation algorithm. *International Journal for Numerical Methods in Engineering*, 39 (9):1475–1489, 1996.

[145] S. A. Canann. Plastering: A new approach to automated, 3D hexahedral mesh generation. *American Institute of Aeronautics and Astronautics*, 1992.

[146] T. D. Blacker and R. J. Meyers. Seams and wedges in plastering: A 3-D hexahedral mesh generation algorithm. *Engineering with Computers*, 9 (2):83–93, 1993.

[147] M. L. Staten, S. J. Owen, and T. D. Blacker. Unconstrained paving & plastering: A new idea for all hexahedral mesh generation. In *Proceedings of the 14th International Meshing Roundtable*, pages 399–416. Springer, 2005.

[148] M. L. Staten, R. A. Kerr, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: progress update. In *Proceedings of the 15th International Meshing Roundtable*, pages 469–486. Springer, 2006.

[149] E. Ruiz-Gironés, X. Roca, and J. Sarrate. The receding front method applied to hexahedral mesh generation of exterior domains. *Engineering with Computers*, 28(4):391–408, 2012.

[150] M. Li and R. Tong. All-hexahedral mesh generation via inside-out advancing front based on harmonic fields. *The Visual Computer*, 28(6-8): 839–847, 2012.

[151] M. S. Ebeida, K. Karamete, E. Mestreau, and S. Dey. Q-TRAN: A new approach to transform triangular meshes into quadrilateral meshes locally. In *Proceedings of the 19th International Meshing Roundtable*, pages 23–34. Springer, 2010.

[152] S. H. Lo. Generating quadrilateral elements on plane and over curved surfaces. *Computers & Structures*, 31(3):421–426, 1989.

[153] B. P. Johnston, J. M. Sullivan, and A. Kwasnik. Automatic conversion of triangular finite element meshes to quadrilateral elements. *International Journal for Numerical Methods in Engineering*, 31(1):67–84, 1991.

[154] C. K. Lee and S. H. Lo. A new scheme for the generation of a graded quadrilateral mesh. *Computers & Structures*, 52(5):847–857, 1994.

[155] S. J. Owen, M. L. Staten, S. A. Canann, and S. Saigal. Q-Morph: An indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 44:1317–1340, 1999.

[156] H. Borouchaki and P. J. Frey. Adaptive triangular-quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 41(5):915–934, 1998.

[157] S. Ramaswami, P. Ramos, and G. Toussaint. Converting triangulations to quadrangulations. *Computational Geometry*, 9(4):257–276, 1998.

[158] C. S. Verma and T. Tautges. Jaal: Engineering a high quality all-quadrilateral mesh generator. In *Proceedings of the 20th International Meshing Roundtable*, pages 511–530. Springer, 2012.

[159] G. Bunin. Non-local topological clean-up. In *Proceedings of the 15th International Meshing Roundtable*, pages 3–20. Springer, 2006.

[160] S. Meshkat and D. Talmor. Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. *International Journal for Numerical Methods in Engineering*, 49(1-2):17–30, 2000.

[161] S. J. Owen and S. Saigal. H-Morph: An indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering*, 49(1-2):289–312, 2000.

[162] S. Yamakawa and K. Shimada. Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. *International Journal for Numerical Methods in Engineering*, 57(15): 2099–2129, 2003.

[163] T. C. Baudouin, J.-F. Remacle, E. Marchandise, F. Henrotte, and C. Geuzaine. A frontal approach to hex-dominant mesh generation. *Advanced Modeling and Simulation in Engineering Sciences*, 1(1):1–30, 2014.

[164] M. L. Staten and K. Shimada. A close look at valences in hexahedral element meshes. *International Journal for Numerical Methods in Engineering*, 83(7):899–914, 2010.

[165] V. D. Liseikin. *Grid generation methods*. Springer Netherlands, 2010.

[166] W. B. Jordan. AEC research and development report. Technical report, KAPL-M-7112, 1970.

[167] A. R. Mitchell, G. Phillips, and E. Wachspress. Forbidden shapes in the finite element method. *IMA Journal of Applied Mathematics*, 8(2): 260–269, 1971.

[168] D. A. Field. Algorithms for determining invertible two-and three-dimensional quadratic isoparametric finite element transformations. *International Journal for Numerical Methods in Engineering*, 19(6):789–802, 1983.

[169] A. E. Frey, C. A. Hall, and T. A. Porsching. Some results on the global inversion of bilinear and quadratic isoparametric finite element transformations. *Mathematics of Computation*, 32(143):725–749, 1978.

[170] D. A. Field. An algorithm for determining invertible quadratic isoparametric finite element transformations. *Mathematics of Computation*, 37: 347–360, 1981.

[171] M. L. Baart and E. J. Mulder. A note on invertible two-dimensional quadratic finite element transformations. *Communications in Applied Numerical Methods*, 3(6):535–539, 1987.

[172] P. M. Knupp. On the invertibility of the isoparametric map. *Computer Methods in Applied Mechanics and Engineering*, 78(3):313–329, 1990.

[173] P. L. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 90(9):1156–1182, 2012.

[174] D. A. Field. Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, 47(4):887–906, 2000.

[175] J. R. Shewchuk. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). Preprint, 2002.

[176] A. Z. I. Salem, S. A. Canann, and S. Saigal. Mid-node admissible spaces for quadratic triangular arbitrarily curved 2D finite elements. *International Journal for Numerical Methods in Engineering*, 50(2):253–272, 2001.

[177] A. Z. I. Salem, S. Saigal, and S. A. Canann. Mid-node admissible space for 3D quadratic tetrahedral finite elements. *Engineering with Computers*, 17(1):39–54, 2001.

[178] P. M. Knupp. Algebraic mesh quality metrics. *SIAM Journal on Scientific Computing*, 23(1):193–218, 2001.

[179] P. M. Knupp. Algebraic mesh quality metrics for unstructured initial meshes. *Finite Elements in Analysis and Design*, 39(3):217–241, 2003.

[180] P. M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. part I—A framework for surface mesh optimization. *International Journal for Numerical Methods in Engineering*, 48(3):401–420, 2000.

[181] P. M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. part II—A framework for volume mesh optimization and the condition number of the jacobian matrix. *International Journal for Numerical Methods in Engineering*, 48(8):1165–1185, 2000.

[182] K.-Y. Yuan, Y.-S. Huang, and T. H. H. Pian. Inverse mapping and distortion measures for quadrilaterals with curved boundaries. *International Journal for Numerical Methods in Engineering*, 37(5):861–875, 1994.

[183] P. Knupp. Label-invariant mesh quality metrics. In *Proceedings of the 18th International Meshing Roundtable*, pages 139–155. Springer, 2009.

[184] X. Roca, A. Gargallo-Peiró, and J. Sarrate. Defining quality measures for high-order planar triangles and curved mesh generation. In *Proceedings of the 20th International Meshing Roundtable*, pages 365–383. Springer, 2012.

[185] A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate. Distortion and quality measures for validating and generating high-order tetrahedral meshes. *Engineering with Computers*, pages 1–15, 2014.

[186] A. Gargallo Peiró. *Validation and generation of curved meshes for high-order unstructured methods*. PhD thesis, Universitat Politècnica de Catalunya, 2014.

[187] A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate. Defining quality measures for mesh optimization on parameterized CAD surfaces. In *Proceedings of the 21st International Meshing Roundtable*, pages 85–102. Springer, 2013.

[188] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

[189] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69:125–130, 1965.

[190] J. Sarrate and A. Huerta. An improved algorithm to smooth graded quadrilateral meshes preserving the prescribed element size. *Communications in Numerical Methods in Engineering*, 17(2):89–99, 2001.

[191] J. Daniels, C. T. Silva, and E. Cohen. Localized quadrilateral coarsening. *Computer Graphics Forum*, 28(5):1437–1444, 2009.

[192] E. Marchandise, C. C. de Wiart, W. G. Vos, C. Geuzaine, and J.-F. Remacle. High-quality surface remeshing using harmonic maps—Part II: Surfaces with high genus and of large aspect ratio. *International Journal for Numerical Methods in Engineering*, 86(11):1303–1321, 2011.

[193] H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 1990.

[194] W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, 1999.

[195] V. Kolmogorov. Blossom V: A new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, 2009.

[196] C. Vande Kerckhove. Génération de maillages hexa-dominants par optimisation combinatoire (in french). Master's thesis, Université Catholique de Louvain, 2013.

[197] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. In *Journal of Machine Learning Research*, pages 503–556, 2005.

[198] T. Jung, L. Wehenkel, D. Ernst, and F. Maes. Optimized look-ahead tree policies: A bridge between look-ahead tree policies and direct policy search. *International Journal of Adaptive Control and Signal Processing*, 28(3-5):255–289, 2014.

[199] B. Lévy and Y. Liu. L_p centroidal Voronoi tessellation and its applications. *ACM Transactions on Graphics (TOG)*, 29(4):119, 2010.

[200] J.-F. Remacle, F. Henrotte, T. Carrier-Baudouin, E. Béchet, E. Marchandise, C. Geuzaine, and T. Mouton. A frontal Delaunay quad mesh generator using the L^∞ norm. *International Journal for Numerical Methods in Engineering*, 94(5):494–512, 2013.

[201] T. C. Baudouin, J.-F. Remacle, E. Marchandise, J. Lambrechts, and F. Henrotte. Lloyd's energy minimization in the L_p norm for quadrilateral surface mesh generation. *Engineering with Computers*, 30(1): 97–110, 2014.

[202] E. Cohen and L. L. Schumaker. Rates of convergence of control polygons. *Computer Aided Geometric Design*, 2(1):229–235, 1985.

[203] A. Johnen. Génération de maillages éléments finis d'ordre élevé (in french). Master's thesis, Université de Liège, 2010.

[204] M. Bergot, G. Cohen, and M. Duruflé. Higher-order finite elements for hybrid meshes using new nodal pyramidal elements. *Journal of Scientific Computing*, 42(3):345–381, 2010.

[205] O. K. Smith. Eigenvalues of a symmetric 3×3 matrix. *Communications of the ACM*, 4(4):168, 1961.

[206] F. Maes, D. L. St-Pierre, and D. Ernst. Monte Carlo search algorithm discovery for single-player games. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(3):201–213, 2013.

APPENDIX A

Paper I: Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm

INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2012; **89**:1102–1119
Published online 6 February 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.3279

Blossom-Quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm

J.-F. Remacle[1,*,†], J. Lambrechts[1], B. Seny[1], E. Marchandise[1], A. Johnen[2] and C. Geuzainet[2]

[1]*Institute of Mechanics, Materials and Civil Engineering (iMMC), Université Catholique de Louvain, Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium*
[2]*Department of Electrical Engineering and Computer Science, Université de Liège, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium*

SUMMARY

A new indirect way of producing all-quad meshes is presented. The method takes advantage of a well-known algorithm of the graph theory, namely the Blossom algorithm, that computes the minimum-cost perfect matching in a graph in polynomial time. The new Blossom-Quad algorithm is compared with standard indirect procedures. Meshes produced by the new approach are better both in terms of element shape and in terms of size field efficiency. Copyright © 2012 John Wiley & Sons, Ltd.

Received 15 December 2010; Revised 13 July 2011; Accepted 13 July 2011

KEY WORDS: quadrilateral meshing; surface remeshing; graph theory; optimization; perfect matching

1. INTRODUCTION

Quadrilateral surface meshes are sometimes considered as superior to triangular meshes for finite element simulations. Discussions about if and why quadrilaterals are better than triangles are usually passionate in the finite element community. We will not try to argue about that thorny question here—but we assume that quadrilateral meshes are indeed useful, and in this paper, we present a new way of generating such meshes.

Let us first briefly recall which kinds of methods can be used to build non-uniform quadrilateral meshes in an automatic manner. There are essentially two categories of methods.

In *direct methods*, the quadrilaterals are constructed at once, either using some kind of advancing front technique [1] or using regular grid-based methods (quadtrees). Advancing front methods for quads are considered to be non-robust, and quadtree methods usually produce low-quality elements close to the boundaries of the domain and are unable to fulfill general size constraints (anisotropy, strong variations).

In *indirect methods*, a triangular mesh is built first. Triangle-merge methods then use the triangles of the initial mesh and recombine them to form quadrangles [2,3]. Other more sophisticated indirect methods use a mix of advancing front and triangle merge [4].

The method we present here is an indirect approach to quadrilateralization. We make use of a famous algorithm of the theory of graphs: the Blossom algorithm, proposed by Edmonds in 1965 [5,6], which allows us to find the minimum-cost perfect matching of a given graph. The new method has some clear advantages: (i) it provides a mesh that is guaranteed to be quadrilateral only; (ii) it is optimal in a certain way; and (iii) it is fast.

*Correspondence to: J.-F. Remacle, Université Catholique de Louvain, Bâtiment Euler, Avenue Georges Lemaître 4, 1348 Louvain-la-Neuve, Belgium.
†E-mail: jean-francois.remacle@uclouvain.be

2. MESH QUALITY MEASURES

The aim of the mesh generation process described in this paper is to build a mesh made of quadrilaterals that has controlled element sizes and shapes. We are interested in generating non-uniform quadrilateral meshes. Local information about sizes is given through the definition of a mesh size field that returns, for every point $\vec{x}$ in the domain, a 'characteristic' length $h(\vec{x})$ that has to be fulfilled by the mesh.

Let $\vec{a}$ and $\vec{b}$ be two points of $\mathbb{R}^3$. The adimensional length of the vector $\overrightarrow{ab}$ with respect to the non-uniform size field h is defined as follows [3, 7]:

$$l^h(\overrightarrow{ab}) = \left\|\overrightarrow{ab}\right\| \int_0^1 \frac{1}{h(\vec{a} + t\,\overrightarrow{ab})}\,\mathrm{d}t. \tag{1}$$

An optimum mesh in terms of the size is a mesh for which every edge i is of adimensional size l_i^h equal to 1. It is of course impossible to have such a perfect unit mesh. Here, we define the *efficiency index* [7] τ of a mesh as the exponential of the mean value of the difference between each edge length l_i^h and 1:

$$\tau[\%] = 100 \exp\left(\frac{1}{n_e}\sum_{i=1}^{n_e} d_i\right), \tag{2}$$

with $d_i = l_i^h - 1$ if $l_i^h < 1$, $d_i = (1/l_i^h) - 1$ if $l_i^h > 1$, and n_e the number of edges in the mesh. Surface mesh algorithms usually produce triangular meshes with typical values of τ around 85%, that is, with non-dimensional sizes around $1/\sqrt{2} \leqslant l_i^h \leqslant \sqrt{2}$.

Having the right sizing for the mesh is not enough: mesh generators should also provide meshes with controlled element qualities. We then define a quality measure for quadrilateral elements. Consider a quadrilateral element q and its four internal angles α_k, $k = 1, 2, 3, 4$. We define the quality $\eta(q)$ of q as follows:

$$\eta(q) = \max\left(1 - \frac{2}{\pi}\max_k\left(\left|\frac{\pi}{2} - \alpha_k\right|\right), 0\right). \tag{3}$$

This quality measure is 1 if the element is a perfect quadrilateral and is 0 if one of those angles is either $\leqslant 0$ or $\geqslant \pi$. In what follows, we will present statistics for the quadrilateral meshes:

- The efficiency index τ, which measures the adequacy of the mesh with the mesh size field. The index τ is smaller or equal to 1 and should be as close as possible to $\tau = 100\%$.
- The average element quality $\bar{\eta}$ as well as the worst element quality η_w, which can be important in the context of finite element simulations.

3. INDIRECT QUADRILATERALIZATION USING A NON-OPTIMAL MATCHING ALGORITHM

In Section 1, we have made the distinction between direct and indirect methods for the construction of quadrilateral meshes. In the case of indirect methods, a triangular mesh is first constructed. Then, triangles are recombined in order to produce quadrangles.

Consider a triangular mesh made of n_t triangles t_i, $i = 1, \ldots n_t$. In what follows, we consider internal edges e_{ij} of the mesh that are common to triangles t_i and t_j. We define a cost function $c(e_{ij}) = 1 - \eta(q_{ij})$ that is associated to each graph edge e_{ij} of the mesh and that is defined as the mesh quality of the quadrilateral q_{ij} that is formed by merging the two adjacent triangles t_i and t_j. Usual indirect quadrilateralization procedures work as follows [3]. Edges e_{ij} of the graph are sorted with respect to their individual cost functions. Then, the two triangles that are adjacent to the best edge e_{ij} of the list are recombined into a quadrilateral. Triangles t_i and t_j are tagged in order to prevent other edges that are adjacent either to t_i or to t_j from being used for another

DOI: 10.1002/nme

quadrilateral forming. Then, the algorithm processes the ordered list of edges, forming quadrilaterals with triangles adjacent to an edge as long as none of those adjacent triangles are tagged. Figure 1 shows an illustration of this procedure for a rectangular domain of size 1×3 and a mesh size field defined by

$$h(x, y) = 0.1 + 0.08 \sin(3x) \cos(6y).$$

Isolated triangles inevitably remain in the mesh, and the resulting mesh is not made of quadrilaterals only. The mesh is then said to be *quad dominant*. In the example of Figure 1, the resulting mesh is made of 836 quads and 240 triangles.

A mesh composed of quadrilaterals can be built subsequently using a uniform mesh refinement procedure [3]. Every quadrilateral of the quad-dominant mesh is split into four sub-quadrilaterals, and every triangle is split into three sub-quadrilaterals (Figure 2). For the size criterion $h(\vec{x})$ to be fulfilled, the initial triangular mesh should thus be built using a size field with twice the value (i.e., $2h$) that is expected in the final mesh.

The recombination process just described is sub-optimal. It does not provide the best set of edges to be recombined with respect to some general cost function. Indeed, the only optimality property of this algorithm is that it ensures that the best triangle pair will be recombined.

The second part of the algorithm, namely the mesh refinement step, also has some drawbacks. Splitting every element of the mesh produces a mesh that has half the size of the initial mesh. It is of course possible to generate an initial mesh with double the required size. Yet, with real geometries, the new vertices will have to be added on the geometry, which is not trivial. On the other hand, the refinement step does not allow a sharp control of the mesh size. On Figure 2, the procedure ends with an efficiency index of 79%, which cannot be considered as good.

In [2], the authors proposed a scheme for recombining triangular meshes that does not always require the refinement step, using a kind of advancing front technique. The merging of triangles starts at the boundary; when a front closes, the algorithm attempts to maintain an even number of triangles on any sub-front. Again, this approach is sub-optimal because the result depends on the ordering of elements and on the choice of the initial front.

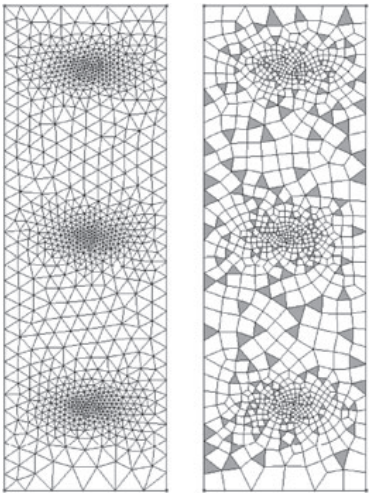


Figure 1. Illustration of the quad-dominant algorithm. The left mesh is the initial triangular mesh, and the right mesh is the quad-dominant mesh, after smoothing (triangles are in gray).

DOI: 10.1002/nme

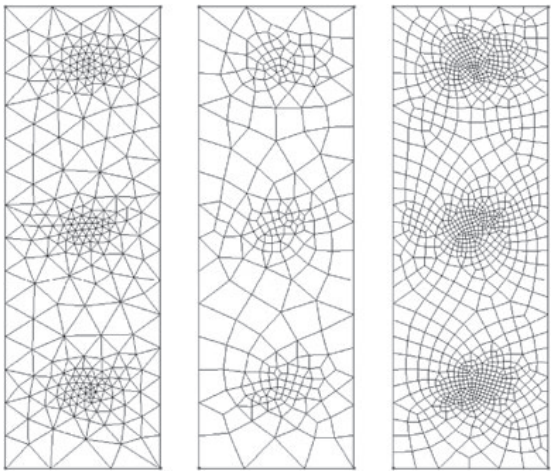


Figure 2. A quad-dominant algorithm using $h^{\text{algo}} = 2h$ followed by a one-mesh refinement procedure leads to a reduction of the efficiency index τ.

4. THE NEW BLOSSOM-QUAD ALGORITHM

Here, our aim is to build a mesh generation scheme that starts with a triangular mesh and attempts to find the set of pairs of triangles that forms the best possible quadrilaterals with the constraint of not leaving any remaining triangle in the mesh.

4.1. Blossom: a minimum-cost perfect-matching algorithm

Let us consider $G(V, E, c)$ as an undirected weighted graph. Here, V is the set of n_V vertices, E is the set of n_E undirected edges, and $c(E) = \sum c(e_{ij})$ is an edge-based cost function, that is, the sum of all weights associated to every edge $e_{ij} \in E$ of the graph. A *matching* is a subset $E' \subseteq E$, such that each node of V has at most one incident edge in E'. A matching is said to be perfect if each node of V has exactly one incident edge in E'. As a consequence, a perfect matching contains exactly $n_{E'} = n_V/2$ edges. A perfect matching can therefore only be found for graphs with an even number of vertices. A matching is optimum if $c(E')$ is minimum among all possible perfect matchings.

In 1965, Edmonds [5, 8] invented the *Blossom algorithm* that solves the problem of optimum perfect matching in polynomial time. A straightforward implementation of Edmonds' algorithm requires $\mathcal{O}(n_V^2 n_E)$ operations.

Since then, the worst-case complexity of the Blossom algorithm has been steadily improving. Both Lawler [9] and Gabow [10] achieved a running time of $\mathcal{O}(n_V^3)$. Galil *et al.* [11] improved it to $\mathcal{O}(n_V n_E \log(n_V))$. The current best-known result in terms of n_V and n_E is $\mathcal{O}(n_V(n_E + \log n_V))$ [12].

There is also a long history of computer implementations of the Blossom algorithm, starting with the Blossom I code of Edmonds *et al.* [6]. In this paper, our implementation makes use of the Blossom IV code of Cook and Rohe [13],[‡] which has been considered for several years as the fastest available implementation of the Blossom algorithm.

[‡]Computer code available at http://www2.isye.gatech.edu/~wcook/blossom4/

DOI: 10.1002/nme

4.2. Optimal triangle merging

Consider now a mesh made of n_t triangles and n_v vertices. Consider a specific weighted graph $G(V, E, c)$ that is built using triangle adjacencies in the mesh. Here, every vertex of the graph is a triangle t_i of the mesh, and every edge of the graph is an internal edge e_{ij} of the mesh that connects two neighboring triangles t_i and t_j. Figure 3 shows a simple triangular mesh with its graph and one perfect matching.

Let us come back first to the non-optimal triangle-merging algorithms of Section 3. In terms of what has just been defined, the subset E' of edges that have been used for triangle merging in the approach of Borouchaki and Frey [3] is a matching that is very rarely perfect. The one of Lee and Lo [2] is usually a perfect matching, but not necessarily the optimal one.

Here, we propose a new indirect approach to quadrilateral meshing that takes advantage of the Blossom algorithm of Edmonds. To this end, we apply the Blossom IV algorithm to the graph of the mesh. We intend to find the optimum perfect matching with respect to the following total cost function:

$$c = \sum_{e \in E'} (1 - \eta(q_{ij})), \tag{4}$$

that is, the sum of all elementary cost functions (or 'badnesses') of the quadrilaterals, which results in the merging of the edges of the perfect matching E'.

An obvious requirement for the final mesh to be quadrilateral only is that the initial triangular mesh contains an even number of triangles (i.e., an even number of graph vertices). Euler's formula for planar triangulations states that the number of triangles in the mesh is

$$n_t = 2(n_v - 1) - n_v^b, \tag{5}$$

where n_v^b is the number of mesh nodes on its boundary. So, the number of mesh points on the boundary n_v^b should be even. Here, our algorithms are applied to general solid models that have a boundary representation [14]. This means that model surfaces are bounded by connected model

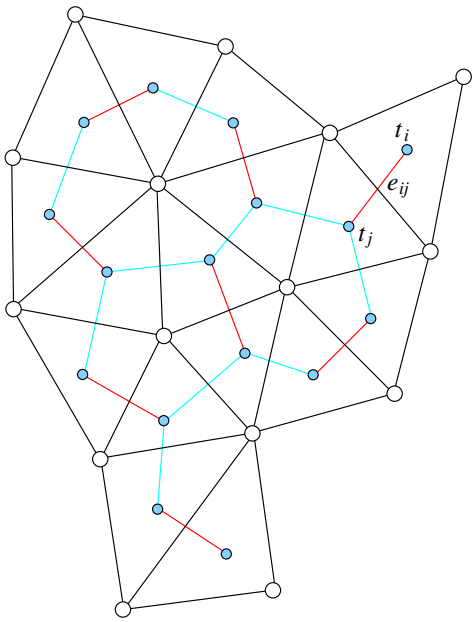


Figure 3. A mesh (in black) and its graph (in cyan and red). The set of graph edges in red forms a perfect matching.

DOI: 10.1002/nme

edges that form edge loops and that the model edges are bounded by model vertices. The mesh vertices of a model edge $n_v^{b_i}$ are defined as the mesh vertices on that edge minus the model vertices. The total number of mesh points on the boundary n_v^b can thus be written as follows:

$$n_v^b = \sum_{i=0}^{N_E} (1 + n_v^{b_i}). \tag{6}$$

It is then easy to see that for n_v^b to be even, it is sufficient for $n_v^{b_i}$ to be odd. This means that a sufficient condition for having an even number of triangles in the mesh is to have every model edge b_i discretized with an odd number of mesh vertices.

Figure 4 shows the same illustrative example as Figures 1 and 2 using the Blossom algorithm for recombining the triangles together with the optimization procedure that will be described in Section 6. The final result is much better not only with respect to the efficiency index (with $\tau = 83\%$) but also with respect to the worst element quality ($\eta_w = 0.405$ instead of $\eta_w = 0.310$ for the mesh of Figure 2). The average quality is better as well.

4.3. Alternative cost functions

Quadrilateral meshes are not isotropic by definition: this is an important difference between quadrilateral and triangular meshes. In principle, the orientation of the mesh can be defined through a 'cross field', that is, a field of orthogonal tangent vectors to the surface. The cross field can be used to orient the edges of the quadrilateralization [15].

It is possible to define an alternative cost function c that is based on the information contained in the cross field. Consider a graph edge e_{ij} that is tangent to the surface with a unit vector $\vec{e}_{ij}$. Assume that the cross field at the midpoint of e_{ij} is defined through two orthogonal unit tangents $\vec{t}_1$ and $\vec{t}_2$. Edges that are aligned with one of the directions of the cross field should not be used for triangle merging. This leads to the following alternative cost function:

$$c(e_{ij}) = \frac{1}{1 - \sqrt{2}/2} \left(\max(|\vec{e}_{ij} \cdot \vec{t}_1|, |\vec{e}_{ij} \cdot \vec{t}_2|) - \sqrt{2}/2 \right). \tag{7}$$

This edge cost function is maximum ($c(e_{ij}) = 1$) for graph edges aligned with one of the directions of the cross field and is minimum ($c(e_{ij}) = 0$) for graph edges that are aligned with the bisector of

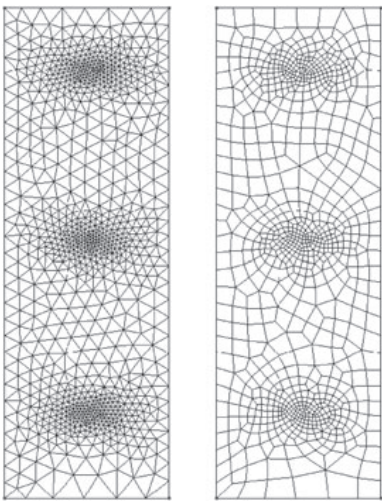


Figure 4. Illustration of the Blossom-Quad algorithm.

DOI: 10.1002/nme

the cross field. For this minimal value of edge cost function, the mesh edges will then be aligned with the orthogonal unit tangents $\vec{t}_1$ and $\vec{t}_2$. Figure 5 compares two meshes of the rectangle with a uniform size field $h = 0.1$. The first mesh makes use of the cost function (4). The elements of this mesh are mainly oriented with the x and y axes. Yet, some patterns of elements are oriented differently. The use of the alternative cost function (7) with a cross field that is aligned with the coordinate axes allows having a quasi-perfect orientation of the mesh. A mix of both cost functions could also be an alternative.

One can use curvature for aligning the quadrilaterals: it is well known that quadrilateral meshes can be aligned in an optimal way for approximating surfaces, depending on the sign of the two eigenvalues of the local curvature tensor. Again, it is possible to find an objective function that takes that into account.

The issue of choosing both an optimal cost function and a suitable cross field is currently under investigation, and we see in these choices many perspectives for further improvements of the Blossom-Quad algorithm.

5. EXISTENCE OF PERFECT MATCHINGS

If for some graphs it is possible to find different perfect matchings, there is in general no guarantee that even one single perfect matching exists in a given graph. Consider the meshes of Figure 6. It is obvious that no perfect matching exists for the coarsest one. The following result, known as Tutte's theorem, proves that none of the two meshes of Figure 6 contains a perfect matching.

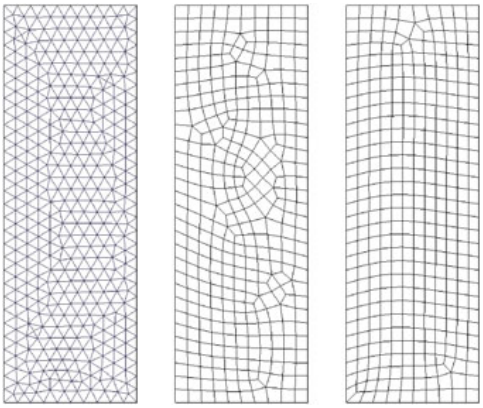


Figure 5. Comparison of meshes generated by Blossom-Quad using cost functions (4) and (7). The initial triangular mesh is on the left. The middle mesh has been generated using (4), and the right mesh uses (7).

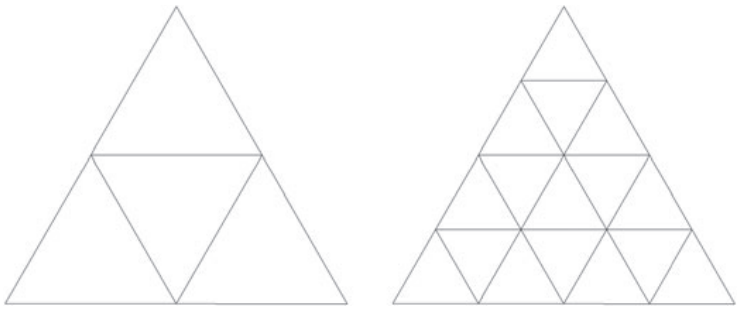


Figure 6. Triangulations that have no perfect matching.

DOI: 10.1002/nme

Tutte's theorem: A graph $G = (V, E)$ has no perfect matching if and only if there is a set $S \subseteq V$ whose removal results in more odd-sized components than the cardinality n_S of S, that is, the number of elements in S [16, 17].

In other words, Tutte's theorem says that there is no perfect matching in a triangulation if and only if it is possible to remove n_S triangles from the mesh and create more n_S non-connected regions that have an odd-sized number of triangles. Let us use Tutte's theorem to prove that none of the two meshes of Figure 6 has a perfect matching. Let us consider the set S of triangles that have their tip pointed downwards. In the coarsest mesh, $n_S = 1$ and 3 odd-sized components are created, which proves that no perfect matching exists. In the second one, six triangles are removed, and 10 odd-sized components are created. This simple pattern can be repeated to produce meshes of arbitrary sizes that have no perfect matchings.

The general problem of counting the number of perfect matchings in a general graph is #P complete.[§] In other words, there is no hope of finding the number of perfect matchings in a general graph. (There is a way to find out, in polynomial time, whether a perfect matching exists by detecting a breakdown in the Blossom algorithm.)

There are however some interesting special cases.

5.1. Planar graphs

A graph is said to be planar if it can be drawn in the 2D plane in such a way that its edges intersect only at its vertices. There exists an efficient algorithm (i.e., in polynomial time) that counts perfect matchings in a planar graph. In planar graphs, graph edges form closed non-overlapping loops that form the graph faces. Let G be a planar graph. Then G can be oriented efficiently so that each face has an odd number of lines oriented clockwise (this orientation is called a Pfaffian orientation of G) [18]. It can be proven that counting the number of perfect matchings can be performed by computing the determinant of the Kasteleyn matrix K:

$$(\#\text{ of perfect matchings of } G)^2 = \det(K), \tag{8}$$

where the Kasteleyn matrix $K(G)$ is an adjacency matrix defined as follows. Consider an edge e_{ij}. If e_{ij} is oriented positively, then $K_{ij} = 1$ and $K_{ji} = -1$. If e_{ij} is oriented negatively, then $K_{ij} = -1$ and $K_{ji} = 1$. If no edge exists between i and j, then $K_{ij} = K_{ji} = 0$.

Here, the computation of the determinant can be carried out in polynomial time so that it is quite easy to count matchings in a triangular mesh. It is therefore possible to compute wether a perfect matching exists in any planar graph. Yet, finding out that no perfect matching exists does not help us a lot at this point. Moreover, the mesh of a whole torus does not lead to a planar graph.[¶]

5.2. Cubic graphs

Cubic graphs, also called trivalent graphs, are graphs for which every node has exactly three adjacent nodes. Every cubic graph has at least one perfect matching [19]. It can be proven that the number of perfect matchings in a cubic graph grows exponentially with n_V.

In a finite element triangulation, most of the triangles of the mesh have three neighbors. Only the triangles that are on the boundary of the domain have less than three neighbors. Thus, in general, a finite element mesh is close to trivalent. We then expect intuitively that perfect matchings will exist in most finite element triangulations. Even though most of the triangulations that we have tried have a perfect matching, the Blossom algorithm has encountered a breakdown for some of the meshes we have tried.

Because cubic graphs always have many perfect matchings, we propose in the Blossom-Quad algorithm to add some extra edges to the graph with the aim of creating a graph topology that is close to trivalent and thus increase the chance of finding perfect matchings.

[§]Sharp P complete, that is, much harder than NP complete.
[¶]The mesh of a complete sphere is planar, even though it is not intuitive.

DOI: 10.1002/nme

5.3. Extra edges

In our approach, we propose to add edges (that we call 'extra edges' in what follows) in the graph of the triangulation in a way that maximizes the chance of existence of a perfect matching. Consider two successive mesh edges on the boundary of the domain. If those edges are like e_1 and e_2 (Figure 7), their neighboring triangles are not connected in the graph. At this point, we add an extra edge for every pair of triangles that are adjacent to those edges that are successive in the boundary of the domain. Those new connections are represented as dotted cyan lines in Figure 7.

Some of those successive mesh edges correspond to triangles that are already connected, like e_3 and e_4 or e_2 and e_5 in Figure 7. This implies that some of the graph nodes have two neighbors; those ones are rounded in red in Figure 7. Some others have four neighbors: those are rounded in green in Figure 7. Yet, there is no node of the graph that has only one adjacent node.

We have not been able to prove that the graph with those extra edges has always a perfect matching. Yet, in our experience, the addition of those extra edges allows us to find a perfect matching in every example that we have tried.

Technically, we assign a high value (typically 1000) to the 'badness' of those extra edges so that an extra edge belongs to the optimal perfect matching only if there exists no perfect matching in the original graph. We propose two manners of post-processing those extra edges when they belong to the perfect matching.

5.3.1. Edge swap. The first algorithm is applied essentially when the extra edge connects two triangles t_1 and t_3 that surround one single triangle t_3 (Figure 8). Edges that are in red in the graph belong to the perfect matching. If e_{13} belongs to the matching, then edge e_{24} belongs to the matching as well. It is therefore possible to swap the mesh edge that connects t_2 and t_4 and build an all-quad configuration. Note that the concave quadrangle that has been created will be removed through topological optimization (Section 6).

5.3.2. Vertex duplication. The second algorithm consists in duplicating the boundary vertex (Figure 9). This algorithm is applied when the two triangles that are connected by the extra edge are surrounded by more than one quad.

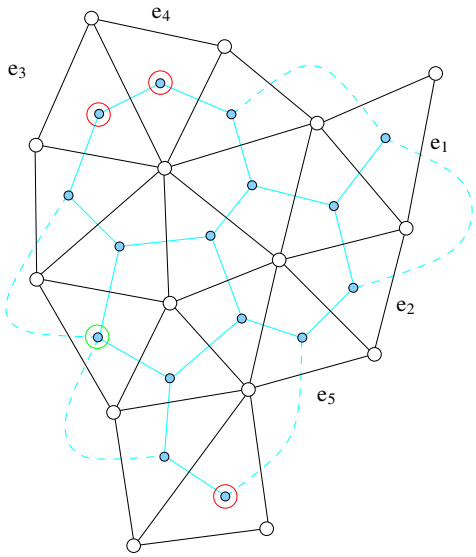


Figure 7. A mesh (in black) and its graph (in cyan). Dotted cyan lines are the extra graph edges that have been added in order to ensure that the graph has perfect matchings.

DOI: 10.1002/nme

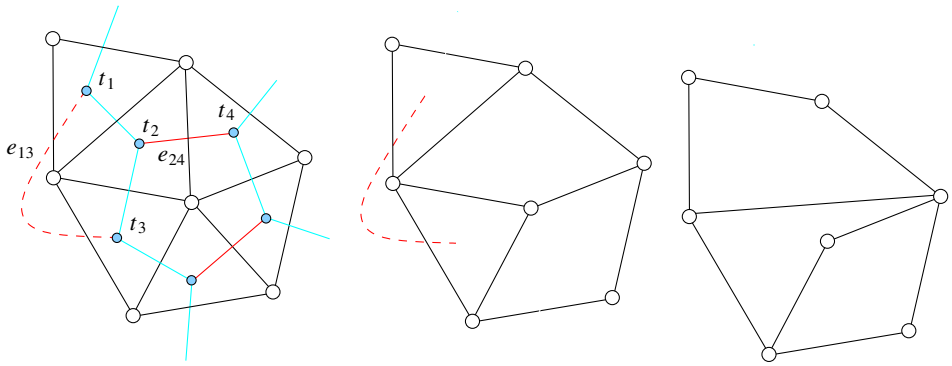


Figure 8. Edge swap algorithm for building an all-quad mesh when an extra edge such as e_{13} belongs to the matching. Example for a configuration with two triangles and two quads.

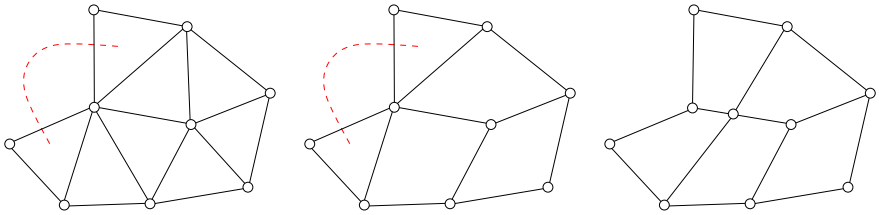


Figure 9. Vertex duplication algorithm for building an all-quad mesh when an extra edge is in the matching.

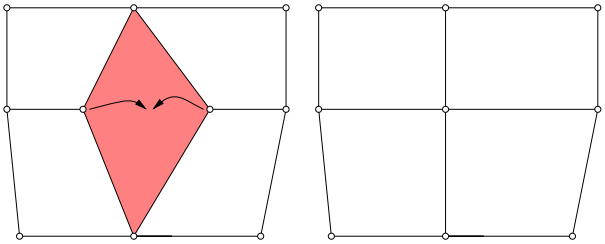


Figure 10. Illustration of quad-vertex-merge optimization operation.

In the next paragraph, we will show how to optimize the quality of the quadrangles of those all-quad meshes using local mesh modifications.

6. OPTIMIZATION

In order to enhance the quality of the final mesh, we first apply a standard vertex smoothing procedure [20] to the nodes of the mesh, taking into account the gradation of the size field. Next, we apply two topological optimization operators specifically tailored for quadrilateral meshes.

The topological optimization operators are local deletion operators: a quad-vertex-merge (Figure 10) and the doublet collapse (Figure 11) operation [21]. These operators allow us to remove local mesh structures that have a bad topology. More precisely, the quad-vertex-merge operator replaces two mesh nodes that have three quadrilateral neighbors by one mesh node with four neighbors, and the doublet collapse removes a vertex that has two neighbors.

DOI: 10.1002/nme

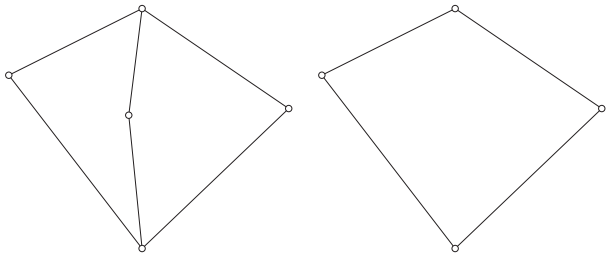


Figure 11. Illustration of a doublet collapse optimization operation.

7. THE BLOSSOM-QUAD ALGORITHM

In this section, we summarize the different steps of the new Blossom-Quad algorithm. This algorithm has been implemented in the open-source mesh generator Gmsh[‖] [14], and examples of how to use it can be found on the Gmsh wiki.[**]

1. Starting from a solid model with a boundary representation, mesh every model edge b_i with an odd number of mesh vertices $n_v^{b_i}$ (6). Then, mesh the model faces with any 2D triangulation algorithm. According to the Euler Equation (5), there will then be an even number of mesh triangles.
2. From the produced mesh, build a weighted graph $G(V, E, c)$ (Figure 3) where the cost function associated to each graph edge $c(e_{ij})$ is given either by (4) or by (7). This weighted graph has then an even number of graph nodes, which is a necessary condition for a perfect matching to exist.
3. Enrich the graph with extra edges such as explained in Section 5.3. Those extra edges are given a very high value of cost function: $c(e_{ij}) = 1000$.
4. Run the Blossom algorithm to find the perfect matching for the given graph.
5. If the perfect matching contains no extra edges, go to step 6. If it contains some, apply the edge swap algorithm and the vertex duplication algorithm (Section 5.3).
6. Optimize the resulting all-quad mesh as explained in Section 6.

Figure 12 shows the global Blossom-Quad procedure applied to an initial triangular mesh.

8. EXAMPLES

In this section, we present the results obtained by applying the new Blossom-Quad algorithm in different contexts. First, we present meshes of simple planar geometries using analytical mesh size fields. Then, we present quadrilateral meshes generated over complex solid models, defined either by a Computer Aided Design (CAD) or StereoLithoGraphy (STL) triangulation. Finally, we show a complex quadrilateral mesh used for multiscale ocean modeling.

8.1. Planar quadrilateral meshes with analytical isotropic size fields

This test case has been proposed by Borouchaki and Frey [3]. The domain is a unit square with a circular hole of radius 0.15 centered at $(0.75, 0.75)$. The mesh size field is taken to have a value of $h(x, y) = 0.003$ along the medial axis of the domain and to have a linear growth from the medial axis to the interior of the domain.

The uniform refinement step that is applied in the recombination algorithm of Borouchaki and Frey [3] has two consequences. On the one hand, it naturally creates a mesh that has a better connectivity. On the other hand, it reduces the efficiency index τ of the mesh. As the recombination

[‖]http://geuz.org/gmsh/

[**]https://geuz.org/trac/gmsh/wiki (username: gmsh and password: gmsh)

DOI: 10.1002/nme

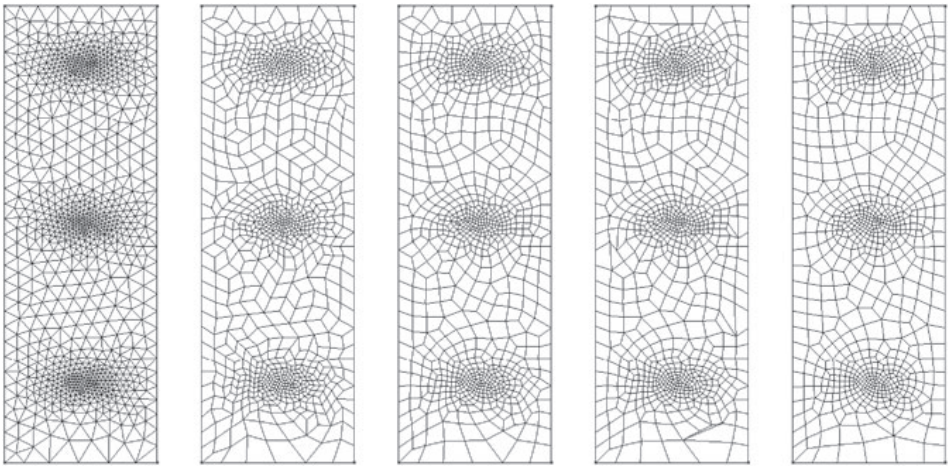


Figure 12. Illustration of the whole Blossom-Quad algorithm.

algorithm of Borouchaki and Frey [3] includes a mesh refinement step, we propose here to generate three meshes with the Blossom-Quad algorithm. The first one applies Blossom to a triangular mesh of size h. The second one applies the Blossom algorithm to a mesh of size $2h$, with one subsequent uniform refinement. The last one applies the Blossom algorithm to a mesh of size $4h$, with two uniform refinements.

Figure 13 compares the quad mesh obtained by Borouchaki and Frey [3] and the three meshes obtained with the presented Blossom-Quad algorithm. For the problem with size h, the Blossom-Quad algorithm takes 1.38 s on a MacBook Pro (Apple Inc., Cupertino, CA, USA) clocked at 2.66 GHz. The new algorithm provides meshes that are of better quality close to the boundaries of the domain. The new algorithm also provides a smoother gradation of the mesh. This is due to the fact that no mesh refinement phase is required in the new algorithm.

Table I compares the quality of the four meshes. We also present some statistics about the degree of the vertices d_i, that is, the number of quads surrounded by a vertex. The new Blossom-Quad algorithm always produces meshes with better element qualities. The application of the Blossom-Quad procedure to the mesh of size h is the best in terms of efficiency τ. Yet, it has less nodes with the optimal topology, that is, with four neighbors. Applying the Blossom-Quad to a mesh of size $2h$ seems to be, at least for this test case, a good compromise. The efficiency is still acceptable, and the quality of the mesh is optimum, both in terms of topology and element quality. Using an initial triangular mesh of size $4h$ does not seem to be a good option, especially in terms of the efficiency τ.

8.2. Quadrilateral mesh generation applied to STL models

In this section, we present quad meshes for complex solid models represented only by a triangulation in STL format.

The first triangulation represents the solid model of the Stanford bunny model, and the second represents a cerebral aneurysm.[††] The latter triangulation is the output of an image segmentation procedure performed from medical data (computed tomography scan). For the quad-remeshing procedure, we first compute an automatic triangular remeshing procedure based on a conformal parametrization as described in [22] and then run the presented Blossom-Quad algorithm.

[††]The STL triangulation of the aneurysm can be found on the National Institute for Research in Computer Science and Control website, http://www-roc.inria.fr/gamma/gamma/download.

DOI: 10.1002/nme

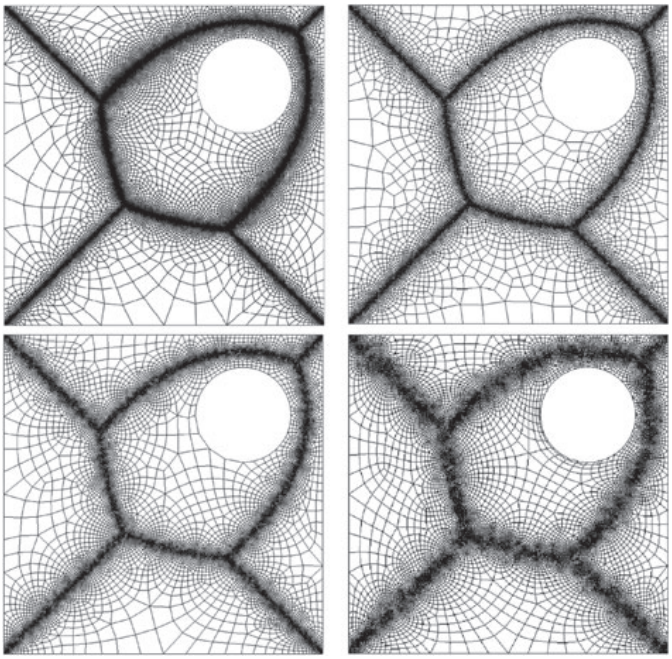


Figure 13. Comparison of both the quadrilateral mesh generation algorithm of Borouchaki and Frey [3] (top, left) and the Blossom-Quad algorithm (other meshes). The top-right mesh has been performed using Blossom on a triangular mesh of size h. The bottom-left mesh has been generated with the Blossom algorithm applied to a triangular mesh of size $2h$, with one subsequent uniform refinement. The last mesh uses the Blossom algorithm on a mesh of size $4h$, with two uniform refinements.

Table I. Quality of the quad meshes for the test case with the medial-axis-based mesh size field h.

	Mesh size	Quad quality		Degree vertices			Efficiency
Algorithm	h^{algo}	η_w	$\bar{\eta}$	$d_4(\%)$	$d_{\min}$	$d_{\max}$	τ
Borouchaki and Frey [3]	$2h$	0.30	0.73	91	3	6	–
Blossom-Quad	h	0.32	0.77	72	3	6	85.6%
Blossom-Quad	$2h$	0.39	0.85	94	3	6	82.3%
Blossom-Quad	$4h$	0.31	0.87	98	3	7	75.4%

The algorithms are run with an initial mesh size h^{algo} that is subsequently refined to reach the prescribed mesh size field h. We present values for the minimum quality η_w, mean quality $\bar{\eta}$, percentage of vertices of degree 4, minimal and maximal values for the degree of vertices, and efficiency index τ.

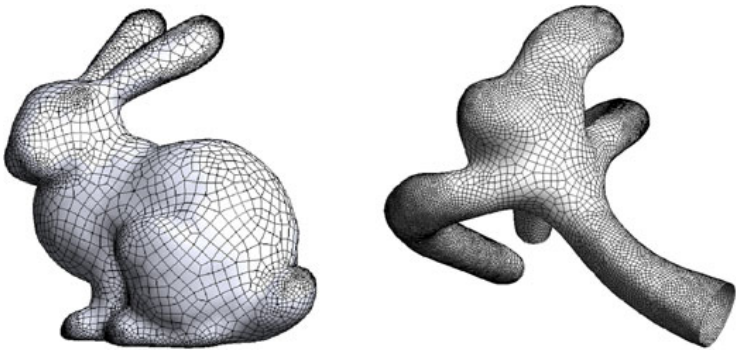
Figure 14 shows two curvature-adapted remeshed STL surfaces, and Figure 15 presents the quality histograms of those meshes. The curvature-adapted meshes are computed by defining the mesh size $h(\vec{x})$ as follows:

$$h(\vec{x}) = \frac{2\pi R(\vec{x})}{N_{\text{p}}}, \quad \text{with } R(\vec{x}) = \frac{1}{\bar{\kappa}(\vec{x})}, \tag{9}$$

where $\bar{\kappa}(\vec{x})$ is the mean curvature that is computed from the initial nodes of the STL triangulation with the algebraic point set surface method (based on the local fitting of algebraic spheres [23]) and N_{p} is the number of points chosen for the radius of curvature ($N_{\text{p}} = 50$).

The overall remeshing procedure for both STL examples takes 21 s: 16 s for the automatic triangular remeshing procedure and only 5 s for the blossom-Quad algorithm. The remeshing was

DOI: 10.1002/nme



$\tau = 84.3\%$ $\tau = 85.6\%$

Figure 14. Isotropic mean-curvature-based quad meshes of 13,000 elements for the STL bunny model (left) and 17,000 elements for the aneurysm model (right).

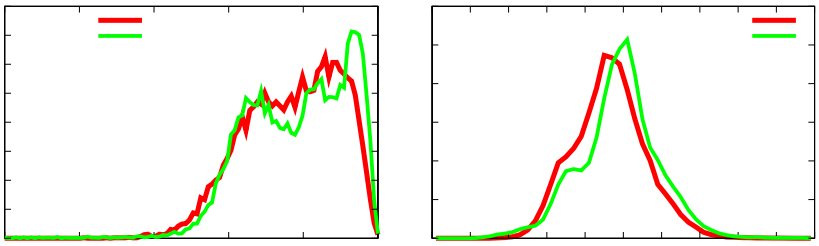


Figure 15. Quality histograms (element quality η (3) and normalized edge length l^h (1)) for the Blossom-Quad remeshing of the bunny and aneurysm models.

performed on a MacBook Pro clocked at 2.66 GHz. This quad-remeshing procedure is extremely fast compared with the quad-dominant meshes of Lévy and Liu [24] for which the remeshing of the Stanford bunny takes 271 s.

8.3. Quadrilateral mesh generation applied to parametric CAD models

We consider the solid model of a human bone. The model contains 11 non-uniform rational B-spline surfaces. The mesh size field is based on surface curvature: we use formula (9) with $N_p = 35$. The total time for surface meshing was 26 s for generating 10,134 quadrilateral elements. The time for computing all 11 perfect matchings and doing optimization does not exceed 10 s. The average quality of the finite element mesh is $\bar{\eta} = 73.3\%$, and the efficiency is $\tau = 80.2\%$. The model still contains one element of bad quality, with $\eta = 0.01$ (Figure 16).

Gmsh allows direct access to the CAD model, allowing us to compute exactly the principal directions of curvature together with minimal and maximal curvature. This allows us to define an anisotropic mesh size field [7]. Here, we use formula (9) with $N_p = 35$ in each of the directions of principal curvature. An anisotropic triangular mesh is initially built using the anisotropic metric provided by the curvature. Then, the Blossom-Quad algorithm is applied to it. The resulting mesh is presented in Figure 17. The quadrilateral mesh naturally aligns itself to the principal directions of curvature, allowing us to build an anisotropic quadrilateral mesh without effort.

8.4. Quadrilateral meshes for ocean modeling

Our research team has developed the first multiscale hydrodynamic model of the whole Great Barrier Reef. The Great Barrier Reef is on the continental shelf of the Australian northeastern coastline and

DOI: 10.1002/nme

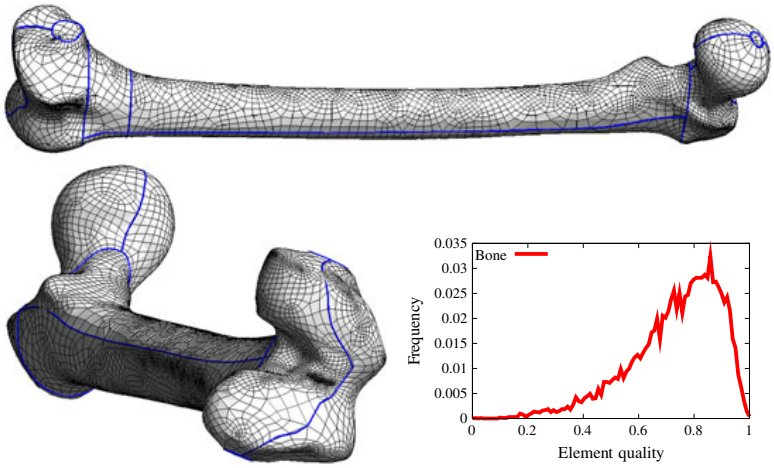


Figure 16. Isotropic quad mesh of the bone geometry with a quality plot.

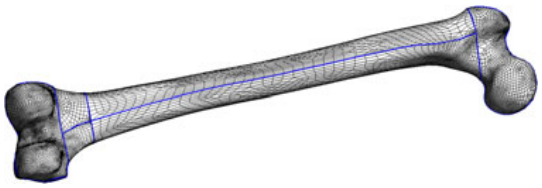


Figure 17. Anisotropic mesh of the bone geometry.

contains over 2500 coral reefs in a strip that is about 2600 km in length and 200 km in width. The mesh size field $h(\vec{x})$ is defined as a function of the bathymetry and the distance to the shore.

We have performed 24 h of simulations of the water circulation on the Great Barrier Reef shelf. The physical model is described in [25]. The equations are discretized with P_1^{DG} discontinuous finite elements combined with a second-order multirate explicit Runge–Kutta temporal integrator as in [26]. A plot of the velocity vectors and the sea surface elevation is presented in Figure 18. Tidal jets and eddies due to the interaction of the flow with the topography near the open-sea boundary are clearly visible. The same simulation on a triangular mesh ($\simeq$250,000 triangles) was 2.7 times slower than on the corresponding Blossom-Quad quadrilateral mesh ($\simeq$119,000 quads).

9. CONCLUSIONS

The main contribution of this paper is to have introduced a well-known result of graph theory (the Blossom algorithm for minimum-cost perfect matching) in the domain of unstructured quadrilateral mesh generation. We have presented a new algorithm—dubbed Blossom-Quad—that takes advantage of this result to produce high-quality quadrilateral meshes in a robust and efficient manner, and we have applied it in different contexts: from planar geometries to parametric CAD models to STL remeshing to multiscale ocean models.

Possible further improvements to the proposed algorithm are numerous. For example, the cost function that has been used could be modified in order to align the quadrilateral mesh with some preferred directions. (We have already presented a simple example to that effect in Section 4.3.) Also, in this paper, we have used triangulations with vertices distributed fairly uniformly as input to the recombination procedure. Recent developments [24, 27] allow us to align those vertices with some prescribed directions. Coupling both approaches could potentially lead to even higher-quality

DOI: 10.1002/nme

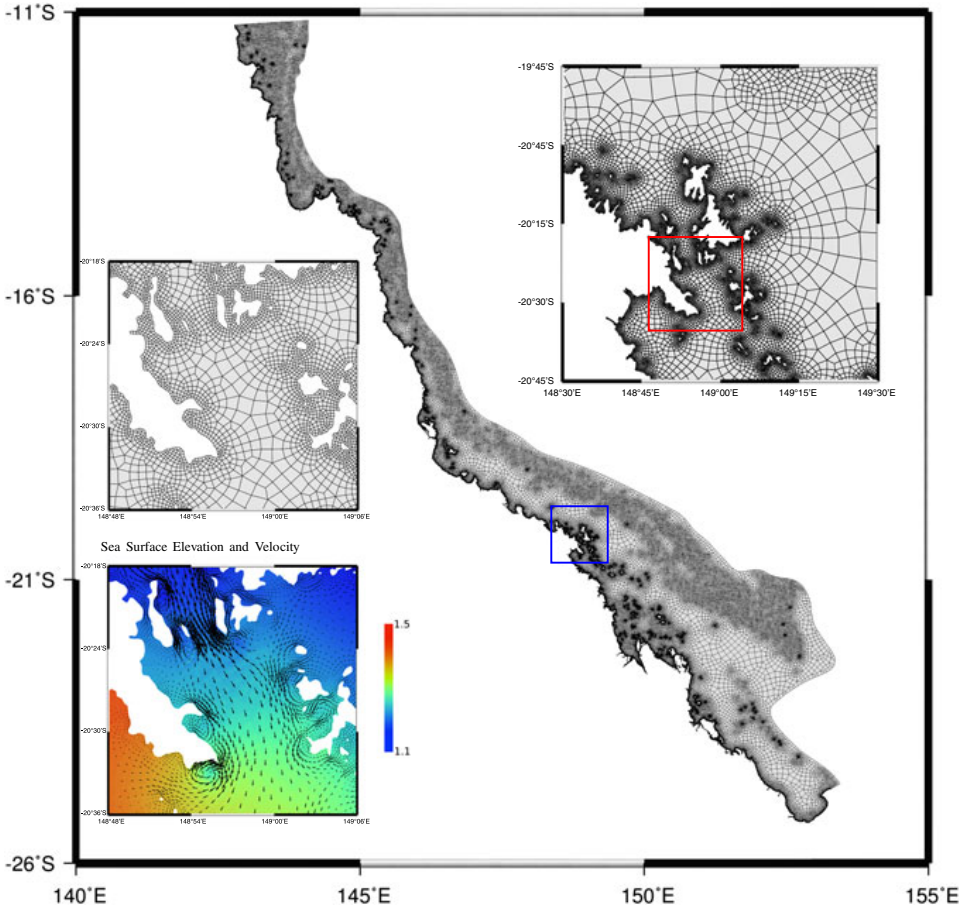


Figure 18. Quadrilateral mesh of the Great Barrier Reef. Two successive zooms of the Whitsunday Islands Archipelago. Sea surface elevation (*color levels*) and bidimensional velocity field (*arrows*).

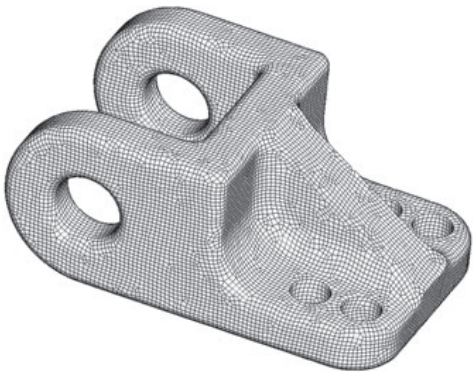


Figure 19. Blossom-Quad algorithm applied to a triangular surface mesh that has been smoothed using the Lp centroidal Voronoï tessellation technique of Lévy and Liu [24].

DOI: 10.1002/nme

quadrilateral meshes. As an example, we have applied the Blossom-Quad algorithm (without optimization) to a triangular surface mesh that has been smoothed using the Lp centroidal Voronoï tessellation technique of Lévy and Liu [24]. The resulting mesh is presented in Figure 19. There, quadrangles do not form patches that are randomly aligned but follow a regular pattern.

In the longer run, the very challenging problem of automatic generation of hexahedral-dominant meshes could be approached using an indirect technique of this kind. In this case, more than two tetrahedra have to be merged in order to form one hexahedron. Here again, we think that graph theory could maybe help us in finding some kind of optimal matching.

ACKNOWLEDGEMENTS

This work has been partially supported by the Belgian Walloon Region under WIST grants ONELAB 1017086 and DOMHEX 1017074.

Authors gratefully thank F. Glineur and J. Hendricks from the Applied Mathematics Department of the Université Catholique de Louvain for the discussions and hints about graph theory.

Authors also acknowledge P. Frey of Université Paris VI for authorizing us to include one of the illustrations of his paper [3] (Figure 13, top left).

Authors finally deeply acknowledge B. Levy of LORIA Nancy for providing us with the triangulation and the smoothing algorithm used to produce Figure 19.

REFERENCES

1. Blacker TD, Stephenson MB. Paving: a new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1991; **32**:811–847.
2. Lee CK, Lo SH. A new scheme for the generation of a graded quadrilateral mesh. *Computers and Structures* 1994; **52**:847–857.
3. Borouchaki H, Frey P. Adaptive triangular–quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering* 1998; **45**(5):915–934.
4. Owen SJ, Staten ML, Canann SA, Saigal S. Q-morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering* 1999; **9**:1317–1340.
5. Edmonds J. Maximum matching and a polyhedron with 0–1 vertices. *Journal of Research of the National Bureau of Standards* 1965; **69B**:125–130.
6. Edmonds J, Johnson EL, Lockhart SC. Blossom I: a computer code for the matching problem. *Report*, IBM T.J. Watson Research Center, Yorktown Heights, New York, 1969.
7. Frey P, George PL. *Mesh Generation—Application to Finite Elements*. Wiley: Hoboken, 2008.
8. Edmonds J. Paths, trees, and flowers. *Canadian Journal of Mathematics* 1965; **17**:449–467.
9. Lawler EL. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart, and Winston: New York, NY, 1976.
10. Gabow H. Implementation of algorithms for maximum matching on nonbipartite graphs. *PhD Thesis*, Stanford University, 1973.
11. Gabow H, Galil Z, Micali S. An o(ev log v) algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing* 1986; **15**:120–130.
12. Gabow HN. Data structures for weighted matching and nearest common ancestors with linking. *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, 1990; 434–443.
13. Cook W, Rohe A. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing* 1999; **11**(2):138–148.
14. Geuzaine C, Remacle JF. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering* 2009; **79**(11):1309–1331.
15. Bommes D, Zimmer H, Kobbelt L. Mixed-integer quadrangulation. *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*. ACM: New York, NY, USA, 2009; 1–10, DOI: 10.1145/1576246.1531383.
16. Tutte WT. A family of cubical graphs. *Proceedings of the Cambridge Philosophical Society* 1947; **43**:459–474.
17. Pemmaraju S, Skiena S. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica* ®. Cambridge University Press: New York, NY, USA, 2003.
18. Kasteleyn PW. Dimer statistics and phase transitions. *Journal of Mathematical Physics* 1963; **4**:287–293. DOI: 10.1063/1.1703953.
19. Oum S. Perfect matchings in claw-free cubic graphs. *ArXiv e-prints*, Jun 2009.
20. Sarrate J, Huerta A. An improved algorithm to smooth graded quadrilateral meshes preserving the prescribed element size. *Communications in Numerical Methods in Engineering* 2001; **17**(2):89–99.
21. Daniels J, II, Silva CT, Cohen E. Localized quadrilateral coarsening. *SGP '09: Proceedings of the Symposium on Geometry Processing*. Eurographics Association: Aire-la-Ville, Switzerland, 2009; 1437–1444.

22. Marchandise E, de Wiart C, Vos W, Geuzaine C, Remacle J. High-quality surface remeshing using harmonic maps—part III: surfaces with high genus and of large aspect ratio. *International Journal for Numerical Methods in Engineering* 2011; **86**:1303–1321.
23. Guennebaud G, Germann M, Gross M. Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum* 2008; **27**:653–662.
24. Lévy B, Liu Y. Lp centroidal Voronoi tessellation and its applications. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, 2010.
25. Lambrechts J, Hanert E, Deleersnijder E, Bernard PE, Legat V, Wolanski JFRE. A high-resolution model of the whole great barrier reef hydrodynamics. *Estuarine, Coastal and Shelf Science* 2008; **79**(1):143–151. DOI: 10.1016/j.ecss.2008.03.016.
26. Constantinescu EM, Sandu A. Multirate timestepping methods for hyperbolic conservation laws. *Journal of Scientific Computing* 2007; **33**:239–278. DOI: 10.1007/s10915-007-9151-y.
27. Hausner A. Simulating decorative mosaics. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, 2001; 573–580.

APPENDIX B

Paper II: Sequential decision-making approach for quadrangular mesh generation

Engineering with Computers
DOI 10.1007/s00366-014-0383-9

ORIGINAL ARTICLE

Sequential decision-making approach for quadrangular mesh generation

Amaury Johnen · Damien Ernst · Christophe Geuzaine

Received: 3 July 2014 / Accepted: 14 October 2014

Abstract A new indirect quadrangular mesh generation algorithm which relies on sequential decision-making techniques to search for optimal triangle recombinations is presented. In contrast to the state-of-art Blossom-quad algorithm, this new algorithm is a good candidate for addressing the 3D problem of recombining tetrahedra into hexahedra.

Keywords Finite element mesh generation · Quadrangular meshes · Recombination · Sequential decision-making

1 Introduction

Finite element methods are numerical techniques largely used for solving physical problems governed by partial differential equations (PDEs). Those methods require the domain of interest to be discretized into a mesh, i.e. a set of discrete subdomains called elements [21]. In 2D, those elements are triangles or quadrilaterals.

Triangular meshes are largely used in finite element simulations because of the existence of fast, robust and automatic techniques to generate high-quality elements, even with size constraints. However, quadrilateral meshes are sometimes considered as superior to triangular meshes [5]. Nonlinear mechanics in particular have element models that only work with quadrangular meshes; in fluid dynamic simulations, quads are also often sought after to discretize boundary layers.

Automatic generation of quad meshes is not an easy problem. For a long time, existing techniques either were complex or produced poor-quality elements (quadrangles with very acute or obtuse angles and/or quadrangles that are poorly aligned with desired directions)—often in areas of the domain where good meshes were critically needed.

At present, there are essentially two approaches for automatically generating quad meshes with size constraints: direct methods and indirect methods. With direct methods, quadrilaterals are constructed at once, either by using advancing front techniques [2], using regular grid-based methods (quadtrees) [6] or partitioning the domain [10]. Indirect methods, on the other hand, rely on an initial triangular mesh (Fig. 1a) and apply merging techniques to recombine the triangles of the initial mesh into quadrangles (Fig. 1b) [3, 11]. Other more sophisticated indirect methods use a mix of advancing front and recombination techniques [16, 20].

The performance of such indirect methods crucially depends on the technique used to generate the initial triangular mesh. Indeed, triangular mesh algorithms usually aim at producing close to equilateral triangles, which is not optimal for recombination [17]. One recent original approach relies on Delaunay-frontal algorithms in L^{∞}-norm to generate close to right-angled triangles, facilitating the construction of high quality quadrangulations [12, 17].

The main shortcoming of classical recombination techniques is the difficulty of generating globally high quality meshes [13, 20]. Greedy recombination of triangles into quads (that merges triangles leading to the best quad first, then continues with the remaining triangles) always leads

A. Johnen · D. Ernst · C. Geuzaine (✉)
Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium
e-mail: cgeuzaine@ulg.ac.be

A. Johnen
e-mail: a.johnen@ulg.ac.be

D. Ernst
e-mail: dernst@ulg.ac.be

Published online: 30 October 2014

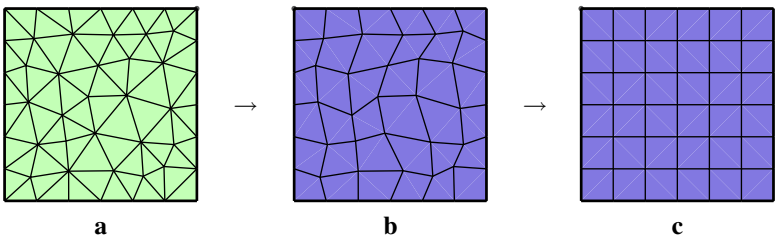


Fig. 1 The *triangles* can be recombined in order to get *quadrangles*. After that, *a* smoothing operation can be applied in order to relocate nodes in better places ($b \rightarrow c$)

to orphaned triangles which have to be subdivided into poor-quality quads. Other techniques using advancing fronts produce poor-quality elements at meeting fronts. Blossom-quad proposed an elegant solution by using a minimum-cost perfect matching algorithm [18]. Blossom-quad guarantees that if a triangle-to-quad recombination exists, it will generate a full quad mesh that is optimal according to a quality functional. However, the best implementation of this algorithm is Blossom V [9], which runs in $O(n_{\triangle}^2)$ where $n_{\triangle}$ is the number of triangles. Moreover, Blossom-quad does not allow for any other "action" than the merging of two triangles into a quad, and it cannot be extended to 3D meshes (to recombine tetrahedra into hexahedra).

In this paper we propose a new indirect method for generating quads. This method can in principle work with any initial triangular mesh, and allows for the use of a great variety of topological and geometrical operations to generate high quality quadrangulations. The time complexity is $O(n_{\triangle})$ and there are a priori no difficulties to extend it to the 3D problem. The conceptual basis of the method relies on looking at the recombination problem as an optimal sequential decision-making problem.

The paper is organized as follows: After stating the problem in Sect. 2, we formulate it as a sequential decision-making problem in Sect. 3. In Sects. 4 and 5, we present two ways of solving the sequential decision-making problem by using look-ahead trees. Finally, the results are presented in Sect. 6 and we conclude in Sect. 7.

2 Problem statement

The goal of the mesh generation process in which the recombination algorithm takes part is, starting from a triangular mesh, to create a mesh made of quadrilaterals that has controlled element sizes and shapes. We assume that the initial triangular mesh fulfills the size requirements. In this paper, we only consider the topological operation of recombining two triangles into one quadrangle and no other topological (swap, collapse) or geometrical (smoothing) operations. This algorithm does not alter the size of edges, so the aim of the recombinations is to produce quadrangles with the best shape quality.

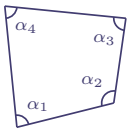


Fig. 2 Quadrilateral with its four internal angles

There exist many different shape quality definitions for quadrangles in the literature. The method we propose is not specific to a particular definition, and we will not enter into the debate about which shape quality measure should be used. In this paper, we will use the definition given in [3]. Consider a quadrilateral element Q and its four internal angles $\alpha_k,\ k = 1, 2, 3, 4$ (Fig. 2). The shape quality $\eta(Q)$ of Q is defined as:

$$\eta(Q) = \max\left(0,\ 1 - \frac{2}{\pi}\max_k\left(\left|\frac{\pi}{2} - \alpha_k\right|\right)\right). \tag{1}$$

This quality measure is equal to 1 if the element is a rectangle and 0 if one of those angles is either ≤ 0 or $\geq \pi$.

We define the global quality q of a mesh $\mathcal{M}$ as the sum of shape quality of every quadrangle:

$$q(\mathcal{M}) = \sum_{Q_i \in \mathcal{Q}} \eta(Q_i), \tag{2}$$

where $\mathcal{Q}$ is the set of quadrangles of $\mathcal{M}$. With this definition, the initial triangular mesh has a global quality of zero.

The problem that we want to solve is the following: from an initial triangular mesh $\mathcal{M}_0$, find, by recombining all triangles into quadrangles, a mesh $\mathcal{M}^*$ that maximizes q, *i.e.* $q(\mathcal{M}^*) \geq q(\mathcal{M}), \forall \mathcal{M}$. Note that we only consider initial triangular meshes that can be fully recombined into quads.

The dual graph of the mesh is the graph $G(V, E)$ for which every vertex v_j is a triangle T_j and every edge e_{ij} represents the adjacency of two neighbour triangles T_i and T_j. It has been shown that the problem of recombining triangles is connected to the problem of finding a matching in the dual graph [18], i.e., a subset of edges that do not

share any vertices. Let us consider a mesh of a closed surface with zero genus, for which every triangle has three neighbours. This is among the worst cases in terms of the ratio between the number of possible recombinations over the number of triangles. The dual graph of such a mesh is a cubic planar graph, *i.e.* all vertices have degree three and it can be drawn on a plane in such a way that its edges do not intersect. In this case it has also been proven that the number of perfect matchings N (matchings which do not leave any unmatched vertex in the graph) grows exponentially with the number of vertices in that graph [7]:

$$3\,\varphi^{n_\triangle/72} \leq N \leq 2^{n_\triangle},$$

where φ is the golden ratio ($\simeq 1.62$). As a result, the number of possible solutions also grows exponentially with the size of the mesh.

Among all the possible solutions, Blossom-quad [18] allows to find the one that maximizes q in polynomial time. However, as stated in the introduction, Blossom-quad cannot be extended to the 3D problem of recombining tetrahedra into hexahedra. Indeed, the formulation of the problem as a minimum-cost perfect matching works in 2D where a pair of triangles forms a quadrangle. In 3D, at least five tetrahedra are required to form a hexahedron. This motivates the formulation of the problem as a sequential decision-making problem.

3 Formulation as a sequential decision-making problem

Recombinations can be seen as actions that are applied step by step to the mesh. This implies that the overall problem can be seen as a discrete-time system for which one seeks a sequence of actions a_t (the recombinations) that maximizes the sum of the rewards $r(a_t)$ defined as the shape quality of the created quadrangles Q_t, i.e. $r(a_t) = \eta(Q_t) \in [0, 1]$. The generic dynamics of this discrete system is given by equation $\mathcal{M}_{t+1} = f(\mathcal{M}_t, a_t)$, where the mesh $\mathcal{M}_t \in X$ is a state of the system and where $a_t \in A(\mathcal{M}_t)$. X is called the state space and $A(\mathcal{M}_t)$ is called the action space. We define $X_f \subset X$ as the set of final states (meshes for which no allowed recombination remains).

Definition 1 The "polder set" containing a triangle T, denoted $\mathcal{P}(T, \mathcal{M})$, is the set of triangles containing at least T for which the dual graph is a connected component of the dual graph of the whole mesh.

Definition 2 Recombination criterion: a recombination between a triangle T and one of its neighbours is forbidden if it separates the set $\mathcal{P}(T, \mathcal{M})$ into two odd-cardinality disjoint polder sets.

In practice, it is easy to understand that an action that leads to a mesh with isolated triangles (or leftover cavities that have an odd number of edges on their boundary) is highly suboptimal. Therefore in the following, we will exclude these actions by taking $A(\mathcal{M}_t)$ as being equal to the set of actions of $\mathcal{M}_t$ that do not violate the recombination criterion (see Definition 2). As a consequence, X is restricted to the set of all meshes obtained by recombining triangles of the initial mesh and for which there is no odd-cardinality *polder set*.

With this formulation, the optimal solution could be found by computing the tree for which every node corresponds to a state $\mathcal{M} \in X$ and every edge corresponds to a possible transition. The leaves of the tree would be the final states and among them we could find the optimal solution. But we showed in Sect. 2 that the size of X_f grows exponentially with the number of initial triangles, and so does the size of the tree. In the next section, we propose an algorithm for navigating into these large trees efficiently. We also relax the original problem statement to allow for non-fully recombined meshes, in which case we consider the percentage of recombinations with respect to the optimal solution as a criterion to evaluate the performance of a the algorithm.

4 Uniform look-ahead tree

The uniform look-ahead tree (LT) can be seen as a computationally efficient way for exploring the tree made of all possible sequences of actions. A uniform LT $\mathcal{T}_h(\mathcal{M}_t)$ expanded from the mesh $\mathcal{M}_t$ with a horizon h is a tree for which the root y_0 is $\mathcal{M}_t$ and every node of depth $d \in \{0, \ldots, h\}$ corresponds to a state that is reachable from $\mathcal{M}_t$ after d transitions (see Fig. 3). Let y_k^{leaf}, $k = 1, 2, \ldots, N_{\text{leaf}}$

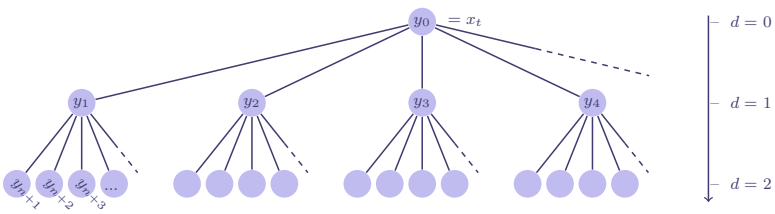


Fig. 3 Uniform look-ahead tree of horizon 2 where n is the number of nodes at depth 1 (i.e. the size of the action space $A(\mathcal{M}_t)$)

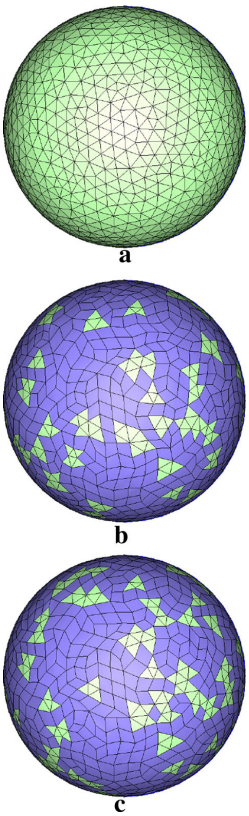


Fig. 4 The triangular mesh of a sphere (**a**) is recombined with a uniform LT of horizon 1 (**b**) and 2 (**c**). The initial mesh contains 2,152 *triangles*. With horizon 1, 1,850 *triangles* are recombined for a quality of 87.6 %. With horizon 2, 1,780 *triangles* are recombined for a quality of 85.1 %

denote the leaf nodes, *i.e.* the nodes that are at depth h. At each leaf node y_k^{leaf} is associated the sequence of h actions $a_\tau^k, \tau = t, \ldots, t+h$. We define the optimal sequence of actions as the one that corresponds to $\max_k \sum_{\tau=t}^{t+h} r(a_\tau^k)$. The action that is applied at time t, a_t, is then taken as the first action of the optimal sequence.

The uniform LT has been applied to the mesh of a sphere that contains 2,152 initial triangles (see Fig. 4). We compute the quality of the final state as a percentage of the quality obtained with Blossom-Quad [18]. For a horizon of 1, we obtain a quality of 87.6 % in 0.16 s. For a horizon of 2, the quality is poorer with 85.1 %, and the computation time is much greater with 225.6 s. Thus the result is worse in both quality and time. Note that on our computer (Macbook Pro Retina, Mid 2012) we have been unable to test the uniform LT for larger horizons.

One explanation for the lack of performance is the fact that with such small horizons the uniform LT algorithm behaves almost similarly to the greedy algorithm that recombines at each step the two triangles leading to the best quad. The fast increase in computation time is explained by the high branching factor, which is of the order of the number of remaining triangles.

In the next section, we define a different way to construct the LT that gives better performances.

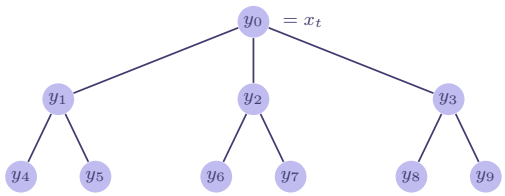


Fig. 5 Selective look-ahead tree of horizon 2. In this example, the *triangle* selected for y_0 has three possible recombinations, so the root has 3 branches. *Triangles* selected at depth 1 have all two possible recombinations

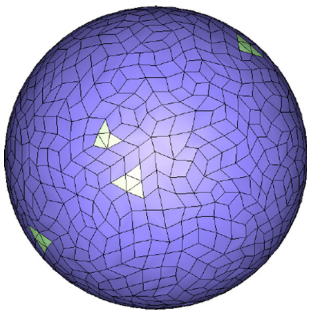


Fig. 6 The triangular mesh of the sphere (Fig. 4) is recombined with the selective LT of horizon 3. On the 2,152 initial triangles, 2,120 are recombined for a quality of 93.8 %

Table 1 The uniform and selective LT are compared on the test case of the sphere

	Horizon	Quality	% recombinations	Time (s)
Uniform	1	87.6	86	0.16
	2	85.1	82.7	225.6
Selective	1	91.8	98.2	0.037
	2	92.8	98.3	0.084
	3	94	98.7	0.146

Results for the selective LT are presented as the median of 100 runs. The quality is presented as a percentage of the quality obtained with Blossom-Quad. A mesh obtained with the selective LT of horizon 3 is showed at Fig. 6

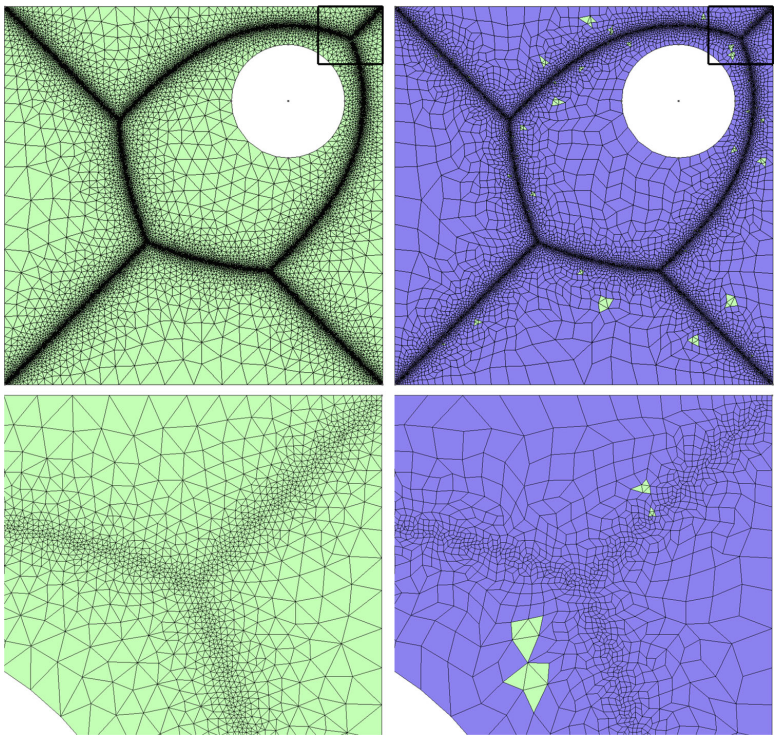


Fig. 7 Borouchaki test: initial mesh and a typical solution of our algorithm with a horizon of 5 (*top*, global mesh, *bottom* zoom of the top right corner). The solution has a quality of 93.17 and 98.8 % of elements are recombined

5 Selective look-ahead tree

The tree made of all possible sequences of actions has intrinsically a large branching factor which hinders the performances of uniform LT techniques. In this section, we propose to apply another form of LT that works with a smaller branching factor allowing to exploit larger depths. We named it *selective look-ahead tree*.

Let us first introduce some definitions. For a given state y, $\mathcal{P}_k(y), k = 1, \ldots, 3$ is the set of all available triangles in y that take part in exactly k recombinations of $A(\mathcal{M}_t)$. We define $\mathcal{P}_{\min}(y)$ as the first non-empty set, i.e. equal to $\mathcal{P}_l(y)$ if $\mathcal{P}_{k<l}(y) = \emptyset$ and $\mathcal{P}_l(y) \neq \emptyset$, or equal to an empty set if $\mathcal{P}_k(y)$ are all empty. The construction of the selective LT is carried out as follows (see Fig. 5):

First, we make the root y_0 to be the current state $\mathcal{M}_t$. Then we randomly select a triangle T_0 that has the minimum of recombinations, i.e. we select T_0 in $\mathcal{P}_{\min}(y_0)$, and we create a child for every possible recombination of T_0. For every child node y_i, we compute $\mathcal{N}_{\triangle}(y_i)$, the set of triangles that can be recombined and are neighbours of the quadrangles created in the sequence from y_0 to y_i. There are then two situations: either $\mathcal{N}_{\triangle}(y_i)$ is empty and T_i is taken in $\mathcal{P}_{\min}(y_0)$ (like the root node) or $\mathcal{N}_{\triangle}(y_i)$ is not empty. For the latter, we compute the first non-empty set of $\mathcal{P}_k(y) \cap \mathcal{N}_{\triangle}(y_i), k = 1, 2$ and randomly select T_i in it. Then, we branch on every possible recombination of T_i.

The rationale behind this is that if the optimal solution corresponds to a full recombination, those selective LT, deployed with the maximum possible depths, would also contain an optimal sequence of actions while still having a much smaller branching factor (bounded by 3 for the root and by 2 for the children). Note that if the optimal solution does not correspond to a fully recombined mesh, it may be reasonable to assume that those new LT may still contain a sequence of actions which is not far from the optimal one.

6 Results

6.1 Mesh of the sphere

We applied the selective LT algorithm to the mesh of the sphere described in Sect. 4. The results are presented at Table 1. We can see that the selective LT shows better

performances than the uniform LT on the three criterions: quality, number of recombinations and computation time. On the other hand, both quality and number of recombinations are increased from horizon 1 to 3.

6.2 Borouchaki mesh

We present the results of the selective LT algorithm obtained for a test case proposed by Borouchaki and Frey [3]. The domain is a unit square with a circular hole of radius 0.15 centred at (0.75, 0.75) with a non-uniform mesh size field (see Fig. 7). The initial triangular mesh has been generated with the delquad algorithm [17] and contains 34,562 triangles (Fig. 7a).

In order to make a comparison with a reference mesh, the triangular mesh has been recombined with the Blossom-quad algorithm using the same quality criterion and without making any additional topological or geometrical optimizations. This is the optimal solution of our search and we take the obtained quality of 10,451.61 as reference. Quality results presented below are all given as percentages of this reference value.

We applied our algorithm for every horizon between 1 and 10. Since triangles are taken randomly, we have, for each horizon, run our algorithm 128 times. Quality, number of recombined elements and computation time are presented as "box and whiskers" graphs in Fig. 8. The first two graphs show that both the quality and the number of recombined elements increase significantly when the horizon increases from 1 to 6. In the third graph, high-horizon values are shown to lead to an average branching factor of $10^{0.49/4} = 1.33$. The asymptotically linear behaviour suggests that the recombination process has a computational complexity of $n_\triangle \alpha^h$, where α is the branching factor.

The computation time (around 1.6 s at horizon 1 and 3.4 s at horizon 2) can be compared to the naive greedy recombination algorithm, which simply recombines triangles by selecting pairs to be recombined in decreasing order of quality of the resulting quads. Without additional constraint, the greedy approach takes 0.98 s to make 15,472 recombinations. Since the remaining triangles are mostly isolated, the quality cannot be compared. The same greedy approach, with the additional constraint of definition 2 (which is equivalent to the uniform LT of Sect. 4 with a horizon equal to 1), is computed in 1.32 s and leads to a quality of 88.5 with 86 % of recombinations. This confirms the usefulness of the sequential decision-making approach, especially in view of its future applicability to 3D problems as well as of its natural handling of additional topological and geometrical actions.

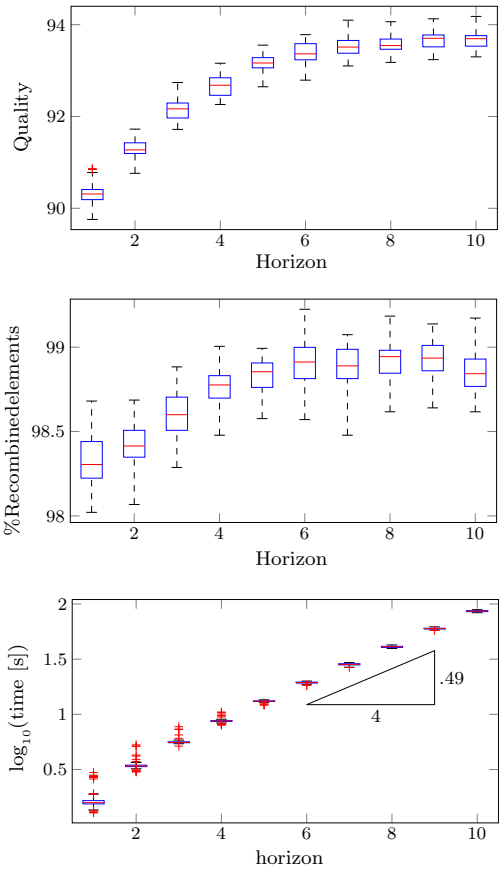


Fig. 8 *Box* and *whiskers* plots for the Borouchaki test, with medians as *red lines*, interquartile ranges (IQR) as *blue boxes*, and *whiskers* in *black*. *Red* crosses depict outliers that are 1.5× IQR above or below the IQR. *Whiskers* only extend to the most extreme data points not considered as outliers. 128 runs have been performed for every horizon from 1 to 10 (colour figure online)

7 Conclusion

We designed a new algorithm to recombine triangular meshes into quadrangular meshes that relies on decision-making techniques. The main advantage of this new approach is that it can produce good quality quad meshes with size constraints in linear time. We showed that the use of selective look-ahead trees instead of uniform look-ahead trees greatly improves the computation time as well as the quality of the final mesh. We also showed that the solution is improved by increasing the horizon of the selective look-ahead trees. The complexity of the algorithm was found

experimentally to be $n_{\triangle}(1+\alpha^h)$, where $\alpha \leq 2$. Unlike Blossom-quad, our algorithm does not guarantee that all triangles are recombined. However, the remaining triangles are grouped and can easily be replaced by quadrangles in post-processing using an adaptation of Bunin's algorithm [4].

Ongoing research focuses on three extensions. First, more advanced tree navigation techniques published in the machine learning literature are investigated [8, 14]. Second, instead of considering topological and geometrical operations (such as edge swaps, collapses or node relocation) as post-processing operations, they could be added directly as actions in the sequential decision-making process. Third, the algorithm could also be extended to recombinations of tetrahedra into hexahedra [1, 15, 19].

Acknowledgments This research project was funded in part by the Walloon Region under WIST 3 grant 1017074 (DOMHEX).

References

1. Baudouin TC, Remacle JF, Marchandise E, Henrotte F, Geuzaine C (2014) A frontal approach to hex-dominant mesh generation. Adv Model Simul Eng Sci 1(1):1–30
2. Blacker TD, Stephenson MB (1991) Paving: a new approach to automated quadrilateral mesh generation. Int J Numer Methods Eng 32(4):811–847
3. Borouchaki H, Frey PJ (1998) Adaptive triangular-quadrilateral mesh generation. Int J Numer Methods Eng 41(5):915–934
4. Bunin G (2006) Non-local topological clean-up. In: Pébay PP (ed) Proceedings of the 15th International Meshing Roundtable. Springer, pp 3–29
5. D'Azevedo E (2000) Are bilinear quadrilaterals better than linear triangles? SIAM J Sci Comput 22(1):198–217
6. Frey PJ, Marechal L (1998) Fast adaptive quadtree mesh generation. In: Proceedings of the Seventh International Meshing Roundtable, pp 211–224
7. Jiménez A, Kiwi M (2011) Counting perfect matchings in the geometric dual. Electron Notes Discret Math 37:225–230
8. Jung T, Wehenkel L, Ernst D, Maes F (2013) Optimized look-ahead tree policies: a bridge between look-ahead tree policies and direct policy search. Int J Adapt Control Signal Process. Available as preprint at: http://onlinelibrary.wiley.com/ doi:10.1002/acs.2387/full
9. Kolmogorov V (2009) Blossom V: a new implementation of a minimum cost perfect matching algorithm. Math Program Comput 1(1):43–67
10. Kowalski N, Ledoux F, Frey P (2013) A PDE based approach to multidomain partitioning and quadrilateral meshing. Springer, pp 137–154
11. Lee CK, Lo SH (1994) A new scheme for the generation of a graded quadrilateral mesh. Comput Struct 52(5):847–857
12. Lévy B, Liu Y (2010) Lp centroidal voronoi tessellation and its applications. ACM Trans Graph 29(4): 119:1–119:11
13. Lo S, Lee C (1992) On using meshes of mixed element types in adaptive finite element analysis. Finite Elem Anal Des 11(4):307–336
14. Maes F, St-Pierre D, Ernst D (2013) Monte carlo search algorithm discovery for single-player games. IEEE Trans Comput Intell AI Games 5(3):201–213. doi:10.1109/TCIAIG.2013.2239295
15. Meshkat S, Talmor D (2000) Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh. Int J Numer Methods Eng 49(1–2):17–30
16. Owen SJ, Staten ML, Canann SA, Saigal S (1999) Q-Morph: an indirect approach to advancing front quad meshing. Int J Numer Methods Eng 44(9):1317–1340
17. Remacle JF, Henrotte F, Carrier-Baudouin T, Bechet E, Marchandise E, Geuzaine C, Mouton T (2013) A frontal delaunay quad mesh generator using the lnorm. Int J Numer Methods Eng 94(5):494–512
18. Remacle JF, Lambrechts J, Seny B, Marchandise E, Johnen A, Geuzaine C (2012) Blossom-quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. Int J Numer Methods Eng 89(9):1102–1119
19. Yamakawa S, Shimada K (2003) Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells. Int J Numer Methods Eng 57(15):2099–2129
20. Zhu J, Zienkiewicz O, Hinton E, Wu J (1991) A new approach to the development of automatic quadrilateral mesh generation. Int J Numer Methods Eng 32(4):849–866
21. Zienkiewicz O, Taylor R (2000) The finite element method—the basis, vol 1. Elsevier

APPENDIX C

Paper III: Geometrical validity of curvilinear finite elements

Journal of Computational Physics 233 (2013) 359–372

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Geometrical validity of curvilinear finite elements

A. Johnen [a], J.-F. Remacle [b], C. Geuzaine [a,*]

[a] Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium
[b] Université catholique de Louvain, Institute of Mechanics, Materials and Civil Engineering (iMMC), Place du Levant 1, 1348 Louvain-la-Neuve, Belgium

ARTICLE INFO

Article history:
Received 23 December 2011
Received in revised form 3 July 2012
Accepted 31 August 2012
Available online 16 September 2012

Keywords:
Finite element method
High-order methods
Mesh generation
Bézier functions

ABSTRACT

In this paper, we describe a way to compute accurate bounds on Jacobian determinants of curvilinear polynomial finite elements. Our condition enables to guarantee that an element is geometrically valid, i.e., that its Jacobian determinant is strictly positive everywhere in its reference domain. It also provides an efficient way to measure the distortion of curvilinear elements. The key feature of the method is to expand the Jacobian determinant using a polynomial basis, built using Bézier functions, that has both properties of boundedness and positivity. Numerical results show the sharpness of our estimates.

1. Introduction

There is a growing consensus in the finite element community that higher-order discretization methods will replace at some point the solvers of today, at least for part of their applications. These high-order methods require a good accuracy of the geometrical discretization to be accurate—in other words, such methods will critically depend on the availability of high-quality curvilinear meshes.

The usual way of building such curvilinear meshes is to first generate a straight sided mesh. Then, mesh entities that are classified on the curved boundaries of the domain are curved accordingly [1–3]. Some internal mesh entities may be curved as well. If we assume that the straight sided mesh is composed of well shaped elements, curving elements introduces a "shape distortion" that should be controlled so that the final curvilinear mesh is also composed of well shaped elements. The optimization of the shape distortion is a computationally expensive operation, especially when applied globally over the full mesh. It is thus crucial to be able to get fast and accurate bounds on the distortion in order to (1) evaluate the quality of the elements during the optimization process; and (2) reduce the sets of elements to be optimized, so that the optimization can be applied locally, i.e., only where it is necessary.

In this paper we present a method to analyze curvilinear meshes in terms of their elementary Jacobian determinants. The method does not deal with the actual generation/optimization of the high order mesh. Instead, it provides an efficient way to guarantee that each curvilinear element is geometrically valid, i.e., that its Jacobian determinant is strictly positive everywhere in its reference domain. It also provides a way to measure the distortion of the curvilinear element. The key feature of the method is to adaptively expand the elementary Jacobian determinants in a polynomial basis that has both properties of boundedness and positivity. Bézier functions are used to generate these bases in a recursive manner. The proposed method can be either used to check the validity and the distortion of an existing curvilinear mesh, or embedded in the curvilinear

* Corresponding author.
E-mail addresses: a.johnen@ulg.ac.be (A. Johnen), jean-francois.remacle@uclouvain.be (J.-F. Remacle), cgeuzaine@ulg.ac.be (C. Geuzaine).

0021-9991/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2012.08.051

mesh generation procedure to assess the validity and the quality of the elements on the fly. The algorithm described in this paper has been implemented in the open source mesh generator Gmsh [4], where it is used in both ways.

2. Curvilinear meshes, distortion and bounds on Jacobian determinants

Let us consider a mesh that consists of a set of *straight-sided* elements of order p. Each element is defined geometrically through its nodes $\mathbf{x}_i,\ i = 1,\dots,N_p$ and a set of Lagrange shape functions $\mathcal{L}_i^{(p)}(\boldsymbol{\xi}),\ i = 1,\dots,N_p$. The Lagrange shape functions (of order p) are based on the nodes $\mathbf{x}_i$ and allow to map a reference unit element onto the real one:

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathcal{L}_i^{(p)}(\boldsymbol{\xi})\, \mathbf{x}_i. \tag{1}$$

The mapping $\mathbf{x}(\xi)$ should be bijective, which means that it should admit an inverse. This implies that the Jacobian determinant $\det \mathbf{x}_{,\xi}$ has to be strictly positive. In all what follows we will always assume that the straight-sided mesh is composed of well-shaped elements, so that the positivity of $\det \mathbf{x}_{,\xi}$ is guaranteed. This standard setting is presented on Fig. 1(left) for the quadratic triangle.

Let us now consider a *curved* element obtained after application of the curvilinear meshing procedure, i.e., after moving some or all of the nodes of the straight-sided element. The nodes of the deformed element are called $\mathbf{X}_i,\ i = 1 \dots N_p$, and we have

$$\mathbf{X}(\boldsymbol{\xi}) = \sum_{i=1}^{N_p} \mathcal{L}_i^{(p)}(\boldsymbol{\xi})\, \mathbf{X}_i. \tag{2}$$

Again, the deformed element is assumed to be valid if and only if the Jacobian determinant $J(\xi) := \det \mathbf{X}_{,\xi}$ is strictly positive everywhere over the ξ reference domain. The Jacobian determinant J, however, is not constant over the reference domain, and computing $J_{\min} := \min_\xi J(\xi)$ is necessary to ensure positivity.

The approach that is commonly used is to sample the Jacobian determinant on a very large number of points. Such a technique is however both expensive and not fully robust since we only get a necessary condition. In this paper we follow a different approach: because the Jacobian determinant J is a polynomial in ξ, J can be interpolated exactly as a linear combination of specific polynomial basis functions over the element. We would then like to obtain provable bounds on $J_{\min}$ by using the properties of these basis functions.

In addition to guaranteeing the geometrical validity of the curvilinear element, we are also interested in quantifying its *distortion*, i.e., the deformation induced by the curving. To this end, let us consider the transformation $\mathbf{X}(\mathbf{x})$ that maps straight sided elements onto curvilinear elements (see Fig. 1). It is possible to write the determinant of this mapping in terms of the ξ coordinates as:

$$\det \mathbf{X}_{,\mathbf{x}} = \frac{\det \mathbf{X}_{,\xi}}{\det \mathbf{x}_{,\xi}} = \frac{J(\boldsymbol{\xi})}{\det \mathbf{x}_{,\xi}}. \tag{3}$$

We call $\mathbf{X}(\mathbf{x})$ the distortion mapping and its determinant $\delta(\xi) := \det \mathbf{X}_{,\mathbf{x}}$ the distortion. The distortion δ should be as close to $\delta = 1$ as possible in order not to degrade the quality of the straight sided element. Elements that have negative distortions are of course invalid but elements that have distortions $\delta \ll 1$ or $\delta \gg 1$ lead to some alteration of the conditioning of the finite element problem. In order to guarantee a reasonable distortion it is thus necessary to find a reliable bound on $J_{\min}$ and $J_{\max} := \max_\xi J(\xi)$ over the whole element.

Note that many different quality measures can be defined based on the Jacobian determinant J. For example, one could look at the Jacobian determinant divided by its average over the element instead of looking at the distortion. Obtaining

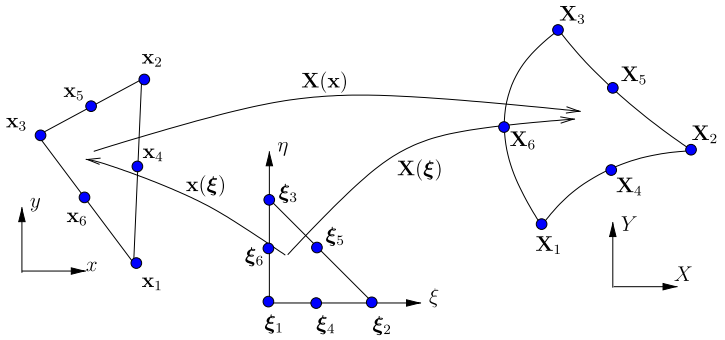


Fig. 1. Reference unit triangle in local coordinates $\xi = (\xi, \eta)$ and the mappings $\mathbf{x}(\xi)$, $\mathbf{X}(\xi)$ and $\mathbf{X}(\mathbf{x})$.

bounds on $J_{\min}$ and $J_{\max}$ is thus still the main underlying challenge. Other interesting choices are presented and analyzed in [5].

3. Bounds for second order planar triangles

We start our analysis with the particular case of second order planar triangles for which a direct computation of $J_{\min}$ is relatively easy. The determinant $J(\boldsymbol{\xi}) = J(\xi,\eta)$ for a planar triangle at order p is a polynomial in ξ and η of order at most $2(p-1)$. For quadratic planar triangles, $J(\xi,\eta)$ is therefore quadratic at most in ξ and η.

The geometry of the six-node quadratic triangle is shown in Fig. 1. Inspection reveals two types of nodes: corners (1, 2 and 3) and midside nodes (4, 5 and 6). If J_i is defined as $J(\xi,\eta)$ evaluated at node i, it is possible to write the Jacobian determinant exactly as a finite element expansion whose coefficients are the Jacobian determinants at the nodes:

$$J(\xi,\eta) = J_1 \underbrace{(1-\xi-\eta)(1-2\xi-2\eta)}_{\mathcal{L}_1^{(2)}(\xi,\eta)} + J_2 \underbrace{\xi(2\xi-1)}_{\mathcal{L}_2^{(2)}(\xi,\eta)} + J_3 \underbrace{\eta(2\eta-1)}_{\mathcal{L}_3^{(2)}(\xi,\eta)} + J_4 \underbrace{4(1-\xi-\eta)\xi}_{\mathcal{L}_4^{(2)}(\xi,\eta)} + J_5 \underbrace{4\xi\eta}_{\mathcal{L}_5^{(2)}(\xi,\eta)} + J_6 \underbrace{4(1-\xi-\eta)\eta}_{\mathcal{L}_6^{(2)}(\xi,\eta)}. \tag{4}$$

In Eq. (4), the functions $\mathcal{L}_i^{(2)}(\xi,\eta)$ are the equidistant quadratic Lagrange shape functions that are commonly used in the finite element community [6].

We first show how to compute the exact minimal Jacobian determinant $J_{\min}$. Then we examine different bounds that can be provided on $J_{\min}$ by exploiting the properties of the basis used in the expansion.

3.1. Exact computation of $J_{\min}$

From Eq. (4), the stationnary point of J can be computed by solving

$$\frac{\partial J}{\partial \xi} = \frac{\partial J}{\partial \eta} = 0, \tag{5}$$

which leads to the following linear system of two equations and two unknowns ξ_{sta} and η_{sta}:

$$\begin{bmatrix} 4(J_1+J_2-2J_4) & 4(J_1-J_4+J_5-J_6) \\ 4(J_1-J_4+J_5-J_6) & 4(J_1+J_3-2J_6) \end{bmatrix} \begin{pmatrix} \xi_{sta} \\ \eta_{sta} \end{pmatrix} = \begin{pmatrix} -(-3J_1-J_2+4J_4) \\ -(-3J_1-J_3+4J_6) \end{pmatrix}. \tag{6}$$

Algorithm 1 allows to compute the minimal Jacobian determinant over one quadratic planar element exactly. If the minimum of the function is outside of the element, it computes the minimum on its border assuming a function $\mathrm{MINQ}(a,b,c)$ that computes

$$\mathrm{MINQ}(a,b,c) = \min_{x\in[0,1]} ax^2 + bx + c. \tag{7}$$

Algorithm 1. Exact computation of J_{min} over a quadratic triangle

```
compute nodal Jacobian determinants J_i, i = 1,...,6;
compute ξ_sta, η_sta as in equation (6);
if η_sta > 0 and ξ_sta > 0 and 1 − ξ_sta − η_sta > 0 then
    J_min = min(J(ξ_sta, η_sta), J_1, J_2, J_3);
else
    m_1 = MINQ(2(J_1 + J_2 − 2J_4), −3J_1 − J_2 + 4J_4, J_1);
    m_2 = MINQ(2(J_1 + J_3 − 2J_6), −3J_1 − J_3 + 4J_6, J_1);
    m_3 = MINQ(2(J_2 + J_3 − 2J_5), −3J_2 − J_3 + 4J_5, J_2);
    J_min = min(m_1, m_2, m_3);
return J_min;
```

Although Algorithm 1 is quite simple, applying similar techniques for higher order elements would become extremely expensive computationally. For example, for a third order triangle, the Jacobian determinant is of order 4 and the algorithm requires the solution of a system of cubic equations; at order 4, it requires the solution of a system of quintic equations. Instead of trying to evaluate $J_{\min}$ directly, we should try to compute (the sharpest possible) bounds in a computationally efficient manner.

3.2. The principle for computing bounds on $J_{\min}$

It is obvious that a necessary condition for having $J(\xi,\eta) > 0$ everywhere is that $J_i > 0,\ i = 1,\dots,6$. Yet, this condition is not sufficient. The expression (4) does not give more information because the quadratic Lagrange shape functions $\mathcal{L}_i^{(2)}(\xi,\eta)$ change sign on the reference triangle. What polynomial basis should we chose to obtain usable bounds?

The first idea is to expand (4) into monomials, which gives:

$$J(\xi,\eta) = J_1 + (-3J_1 - J_2 + 4J_4)\xi + (-3J_1 - J_3 + 4J_6)\eta + 4(J_1 - J_4 + J_5 - J_6)\xi\eta + 2(J_1 + J_2 - 2J_4)\xi^2 + 2(J_1 + J_3 - 2J_6)\eta^2. \tag{8}$$

Every monomial being positive on the reference triangle, we have now a set of sufficient conditions that can be written as

$$4J_4 \geqslant 3J_1 + J_2, \quad 4J_6 \geqslant 3J_1 + J_3, \quad J_1 + J_5 \geqslant J_4 + J_6, \quad J_1 + J_2 \geqslant 2J_4, \quad J_1 + J_3 \geqslant 2J_6.$$

However these constraints do not provide a usable bound on $J_{\min}$ and break the symmetry of the expression with respect to a rotation of corner nodes.

A second idea is to expand (4) in terms of the second order hierarchical basis functions $\psi_i(\xi,\eta),\ i = 1,\dots,6$, which are also positive on the triangle [7]:

$$\begin{aligned} J(\xi,\eta) = {} & J_1 \underbrace{(1-\xi-\eta)}_{\psi_1(\xi,\eta)} + J_2 \underbrace{\xi}_{\psi_2(\xi,\eta)} + J_3 \underbrace{\eta}_{\psi_3(\xi,\eta)} + (4J_4 - 2J_1 - 2J_2) \underbrace{(1-\xi-\eta)\xi}_{\psi_4(\xi,\eta)} + (4J_5 - 2J_3 - 2J_2) \underbrace{\xi\eta}_{\psi_5(\xi,\eta)} \\ & + (4J_6 - 2J_1 - 2J_3) \underbrace{(1-\xi-\eta)\eta}_{\psi_6(\xi,\eta)}. \end{aligned} \tag{9}$$

This last expression has the right symmetry, and leads to the following validity conditions:

$$J_1 \geqslant 0, \quad J_2 \geqslant 0, \quad J_3 \geqslant 0, \quad 4J_4 \geqslant 2J_1 + 2J_2, \quad 4J_5 \geqslant 2J_2 + 2J_3, \quad 4J_6 \geqslant 2J_3 + 2J_1. \tag{10}$$

$J(\xi,\eta)$ is a degree two polynomial, therefore it has an expression in the (Ψ_i) basis. Let K_i denote the coefficients in this basis. Writing $J(\xi,\eta) := \sum_{i=1}^{6} \psi_i(\xi,\eta) K_i$, we have

$$\min_{\xi,\eta} J(\xi,\eta) = \min_{\xi,\eta}\left(\sum_i \psi_i(\xi,\eta) K_i\right) \geqslant \min_{\xi,\eta}\left(\sum_i \psi_i(\xi,\eta)\right) \min_i K_i = \min_i K_i,$$

because $\sum_i \psi_i = 1 + \xi + \eta - \xi^2 - \eta^2 - \xi\eta$ has its minimum on the corner nodes (where its value is equal to 1). And since $K_i,\ i = 1,\dots,3$ are values of the Jacobian determinant (at the three corners), they form an upper bound on it. Thus, expansion (9) leads to the following estimate for the minimum of the Jacobian determinant over the triangle:

$$\min(J_1, J_2, J_3, 4J_4 - 2J_1 - 2J_2, 4J_5 - 2J_2 - 2J_3, 4J_6 - 2J_3 - 2J_1) <= J_{\min} <= \min(J_1, J_2, J_3). \tag{11}$$

It is easy to see that the estimate is however of very poor quality: for an element that has a constant and positive J, (11) simply tells us that $J_{\min} \geqslant 0$.

In order to find a sharper estimate, instead of the hierarchical quadratic functions $\psi_i(\xi,\eta)$, we can use the quadratic triangular Bézier functions $\mathcal{B}_2^{(2)}(\xi,\eta)$ [8]:

$$\begin{aligned} J(\xi,\eta) = {} & J_1 \underbrace{(1-\xi-\eta)^2}_{\mathcal{B}_1^{(2)}(\xi,\eta)} + J_2 \underbrace{\xi^2}_{\mathcal{B}_2^{(2)}(\xi,\eta)} + J_3 \underbrace{\eta^2}_{\mathcal{B}_3^{(2)}(\xi,\eta)} + \left(2J_4 - \frac{1}{2}(J_2 + J_1)\right) \underbrace{2\xi(1-\xi-\eta)}_{\mathcal{B}_4^{(2)}(\xi,\eta)} + \left(2J_5 - \frac{1}{2}(J_3 + J_2)\right) \underbrace{2\xi\eta}_{\mathcal{B}_5^{(2)}(\xi,\eta)} \\ & + \left(2J_6 - \frac{1}{2}(J_1 + J_3)\right) \underbrace{2\eta(1-\xi-\eta)}_{\mathcal{B}_6^{(2)}(\xi,\eta)}. \end{aligned} \tag{12}$$

Since $\sum_{i=1}^{6} \mathcal{B}_i^{(2)}(\xi,\eta) = 1$ and $\mathcal{B}_i^{(2)}(\xi,\eta) \geqslant 0$, we obtain the following estimate

$$\min\left(J_1, J_2, J_3, 2J_4 - \frac{J_1+J_2}{2}, 2J_5 - \frac{J_2+J_3}{2}, 2J_6 - \frac{J_3+J_1}{2}\right) <= J_{\min} <= \min\{J_1, J_2, J_3\}. \tag{13}$$

One can show that this estimate is always better than the one using the hierarchical basis. It provides two conditions on the geometrical validity of the triangle: a *sufficient* condition (if $\min\left\{J_1, J_2, J_3, 2J_4 - \frac{J_1+J_2}{2}, 2J_5 - \frac{J_2+J_3}{2}, 2J_6 - \frac{J_3+J_1}{2}\right\} > 0$, the element is valid) and a *necessary* condition (if $\min\{J_1, J_2, J_3\} < 0$, the element is invalid). However, these two conditions are sometimes insufficient to determine the validity of the element, as the bound (13) is often not sharp enough (having $\min\left\{2J_4 - \frac{J_1+J_2}{2}, 2J_5 - \frac{J_2+J_3}{2}, 2J_6 - \frac{J_3+J_1}{2}\right\} < 0$ does not imply that the element is invalid).

A sharp necessary and sufficient condition on the geometrical validity of an element can be achieved in a general way by refining the Bézier estimate adaptively so as to achieve any prescribed tolerance—and thus provide bounds as sharp as necessary for a given application.

4. Adaptive bounds for arbitrary curvilinear finite elements

In order to explain the adaptive bound computation let us first focus on the one-dimensional case, for "line" finite elements. Since Bézier functions can be generated for all types of common elements (triangles, quadrangles, tetrehedra, hexahedra and prisms), the generalization to 2D and 3D elements will be straightforward.

4.1. The one-dimensional case

In 1D the Bézier functions are the Bernstein polynomials:

$$\mathcal{B}_k^{(n)}(\xi) = \binom{n}{k}(1-\xi)^{n-k}\,\xi^k \quad (\xi \in [0,1];\ k = 0,\dots,n), \tag{14}$$

where $\binom{n}{k} = \frac{n!}{n!(n-k)!}$ is the binomial coefficient. The Bézier interpolation requires $n+1$ control values b_i. We have

$$J(\xi) = \sum_{k=0}^{N_n} \mathcal{B}_k^{(n)}(\xi)\, b_k. \tag{15}$$

Bernstein–Bézier functions have the nice following properties: (i) they form a partition of unity which means that $\sum_{k=0}^{n}\mathcal{B}_k^{(n)}(\xi) = 1$ for all $\xi \in [0,1]$ and (ii) they are positive which means that $\mathcal{B}_k^{(n)}(\xi) \geqslant 0$ for all $\xi \in [0,1]$. This leads to the well known property of Bézier interpolations:

$$\min_{\xi\in[0,1]} J(\xi) \geqslant b_{\min} = \min_i b_i \quad \text{and} \quad \max_{\xi\in[0,1]} J(\xi) \leqslant b_{\max} = \max_i b_i. \tag{16}$$

Moreover, the control values related to the corner nodes of the element are equal to the values of the interpolated function. In what follows we assume that these "corner" control values are always ordered at the K_f first indices. We then have

$$\min_{\xi\in[0,1]} J(\xi) \leqslant \min_{i<K_f} b_i \quad \text{and} \quad \max_{\xi\in[0,1]} J(\xi) \geqslant \max_{i<K_f} b_i. \tag{17}$$

Since Lagrange and Bézier functions span the same function space, computation of the Bézier values b_i from the nodal values J_i (and convertly) is done by a transformation matrix. The tranformation matrix $\boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}$, which computes nodal values from control values, is created by evaluating Bézier functions at sampling points:

$$\boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}} = \begin{bmatrix} \mathcal{B}_0^{(n)}(\xi_0) & \dots & \mathcal{B}_n^{(n)}(\xi_0) \\ \mathcal{B}_0^{(n)}(\xi_1) & \dots & \mathcal{B}_n^{(n)}(\xi_1) \\ \vdots & \ddots & \vdots \\ \mathcal{B}_0^{(n)}(\xi_n) & \dots & \mathcal{B}_n^{(n)}(\xi_n) \end{bmatrix}.$$

Those sampling points are taken uniformly, i.e., at the location of the nodes of the element of order n. The inverse transformation is $\boldsymbol{T}^{(n)}_{\mathcal{L}\to\mathcal{B}} = \boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}{}^{-1}$ and from the expression of the interpolation of the Jacobian determinant (15), we can write

$$J = \boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}} B$$

$$B = \boldsymbol{T}^{(n)}_{\mathcal{L}\to\mathcal{B}} J, \tag{18}$$

where B and J are the vectors containing respectively the b_i's and the J_i's.

4.2. Adaptive subdivision

Let us assume that the domain $[0,1]$ is subdivided into Q parts. The interpolation $J^{[q]}(\xi^{[q]})$ on the qth subdomain $[a,b]$ $(0 \leqslant a < b \leqslant 1)$ must verify

$$J^{[q]}(\xi^{[q]}) = \sum_{k=0}^{N_n} \mathcal{B}_k^{(n)}(\xi^{[q]})\, b_k^{[q]} = \sum_{k=0}^{N_n} \mathcal{B}_k^{(n)}(\xi(\xi^{[q]}))\, b_k \quad (\xi^{[q]} \in [0,1]), \tag{19}$$

with $\xi(\xi^{[q]}) = a + (b-a)\,\xi^{[q]}$.

Considering the nodes $\xi_k^{[q]}$ such that $\xi_k^{[q]} = \xi_k$ $(k = 0,\dots,n)$ (i.e., such that they are ordered like the sampling points), the expression (19) reads

$$\boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}} B^{[q]} = \begin{bmatrix} \mathcal{B}_0^{(n)}(a+(b-a)\xi_0) & \dots & \mathcal{B}_n^{(n)}(a+(b-a)\xi_0) \\ \mathcal{B}_0^{(n)}(a+(b-a)\xi_1) & \dots & \mathcal{B}_n^{(n)}(a+(b-a)\xi_1) \\ \vdots & \ddots & \vdots \\ \mathcal{B}_0^{(n)}(a+(b-a)\xi_n) & \dots & \mathcal{B}_n^{(n)}(a+(b-a)\xi_n) \end{bmatrix} B = \boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}{}^{[q]} B,$$

where $B^{[q]}$ is the vector containing the control values of the qth subdomain. This implies that

$$B^{[q]} = \left[\boldsymbol{T}^{(n)}_{\mathcal{L}\to\mathcal{B}}\boldsymbol{T}^{(n)}_{\mathcal{B}\to\mathcal{L}}{}^{[q]}\right] B = \mathbf{M}^{[q]} B. \tag{20}$$

Each set of new control values bounds the Jacobian determinant on its own subdomain and we have:

$$b'_{\min} = \min_{i,q} b_i^{[q]} \leqslant J_{\min} \leqslant \min_{i<K_f,q} b_i^{[q]} \tag{21}$$

and

$$\max_{i<K_f,q} b_i^{[q]} \leqslant J_{\max} \leqslant b'_{\max} = \max_{i,q} b_i^{[q]}. \tag{22}$$

If an estimate is not sufficiently sharp, we can thus simply subdivide the appropriate parts of the element. This leads to a simple adaptive algorithm, exemplified in Fig. 2. In this particular case the original estimate (16) and (17) is not sharp enough ($J_{\min} \in [-3, 1]$). After one subdivision, the Jacobian determinant is proved to be positive on the second subdomain. The first subdomain is thus subdivided once more, which proves the validity. In practice, as will be seen in Section 5, a few levels of refinement lead to the desired accuracy. The subdivision has quadratic speed of convergence [9,10].

Note that in a practical implementation (in finite precision arithmetic) we must take care of a tricky situation. If the minimum of the Jacobian determinant is too close to zero but positive, then the upper bound is positive while the lower bound might never get positive. In order to avoid this situation, we limit the number of consecutive subdivisions that can be applied. The undetermined elements are then considered as invalid. Another way of getting rid of this issue is to relax the condition of rejection as explained in Section 4.4.

4.3. Extension to higher dimensions

The extension of the method to higher dimensions is straightforward, provided that Bézier functions can be generated and that a subdivision scheme is available. Jacobian determinants J are polynomials of $\xi,\ \eta$ in 2D and of $\xi,\ \eta,\ \zeta$ in 3D.

For high order triangles, the Bézier triangular polynomials are defined as

$$\mathcal{T}_{i,j}^{(p)}(\xi,\eta) = \binom{p}{i}\binom{p-i}{j}\ \xi^i\ \eta^j\ (1-\xi-\eta)^{p-i-j} \quad (i+j \leqslant p).$$

It is possible to interpolate any polynomial function of order at most p on the unit triangle $\xi > 0,\ \eta > 0,\ \xi+\eta < 1$ as an expansion into Bézier triangular polynomials. Recalling that, for a triangle at order p, its Jacobian determinant $J(\xi,\eta)$ is a polynomial in ξ and η at order at most $n = 2(p-1)$, we can write

$$J(\xi,\eta) = \sum_{i+j\leqslant n} b_{ij}\mathcal{T}_{i,j}^{(n)}(\xi,\eta).$$

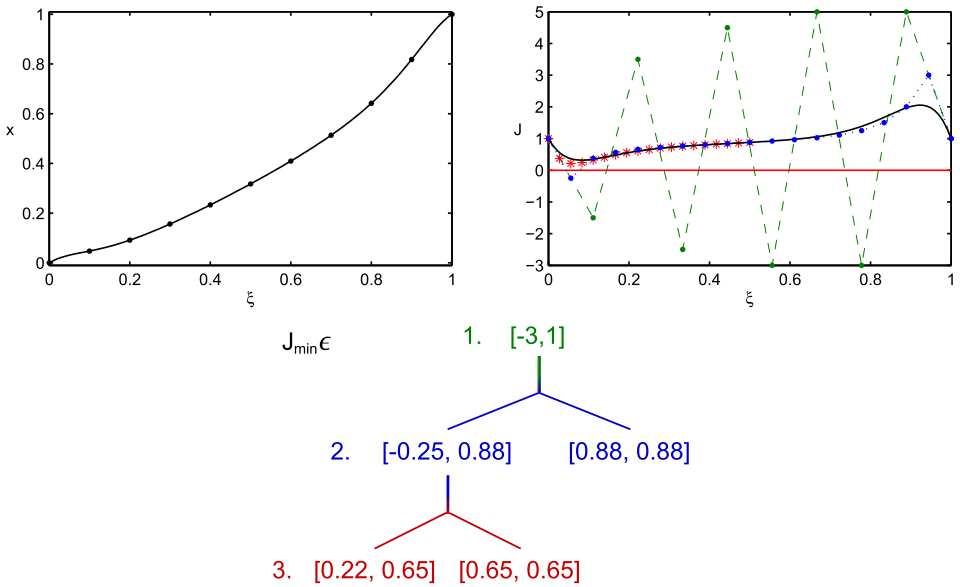


Fig. 2. Top left: one-dimensional element mapping $x(\xi)$. Top right: Exact Jacobian determinant $J(\xi)$ (black), control values on the original control points (green) and two adaptive subdivisions (blue and red). Bottom: estimates of $J_{\min}$ at each step in the adaptive subdivision process. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Order of the Jacobian determinant and number of coefficients in the expansion for an element of order p.

	Order (n) of J	Number of coefficients
Line	$p-1$	$n+1$
Triangle	$2(p-1)$	$(n+1)(n+2)/2$
Quadrangle	$2p-1$	$(n+1)^2$
Tetrahedron	$3(p-1)$	$(n+1)(n+2)(n+3)/6$
Prism	$3p-1$	$(n+1)^2(n+2)/2$
Hexahedron	$3p-1$	$(n+1)^3$

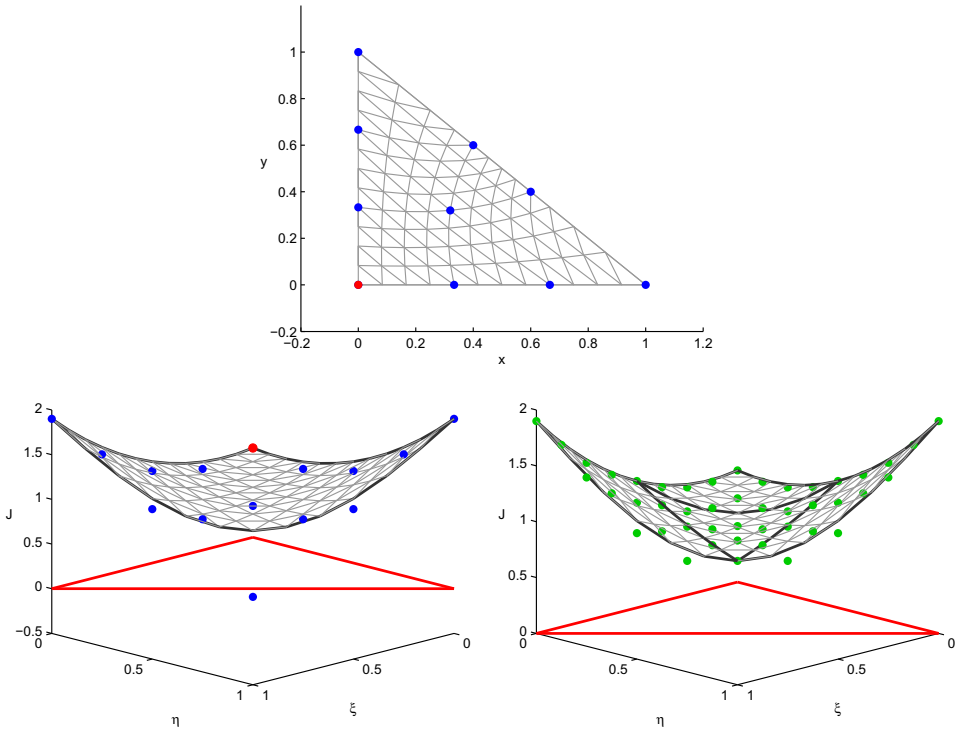


Fig. 3. Top: third-order planar triangle. Bottom left: exact Jacobian determinant and control values (dots) on the original control points; the validity of the element cannot be asserted. Bottom right: exact Jacobian determinant and control values (dots) after one subdivision; the element is provably correct.

It is also possible to compute J in terms of Lagrange polynomials

$$J(\xi,\eta) = \sum_i J_i \mathcal{L}_i^{(n)}(\xi,\eta),$$

where the J_i are the Jacobian determinants calculated at Lagrange points. It is then easy to find a transformation matrix $T^n_{\mathcal{LB}}$ such that

$$B = T^n_{\mathcal{LB}} J,$$

where B and J are the vectors containing respectively the control values of the Jacobian determinant b_{ij} and the J_i's. As an example, for quadratic triangles we obtain

$$T^2_{\mathcal{LB}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -1/2 & -1/2 & 0 & 2 & 0 & 0 \\ 0 & -1/2 & -1/2 & 0 & 2 & 0 \\ -1/2 & 0 & -1/2 & 0 & 0 & 2 \end{pmatrix}, \tag{23}$$

which directly provides the estimate (13).

Other element shapes can be treated similarly. For quadrangles, tetrahedra, prisms and hexahedra, the Bézier are functions respectively:

$$\mathcal{Q}^{(p)}_{i,j}(\xi,\eta) = \mathcal{B}^{(p)}_i(\xi)\ \mathcal{B}^{(p)}_j(\eta) \quad (i \leqslant p, j \leqslant p),$$

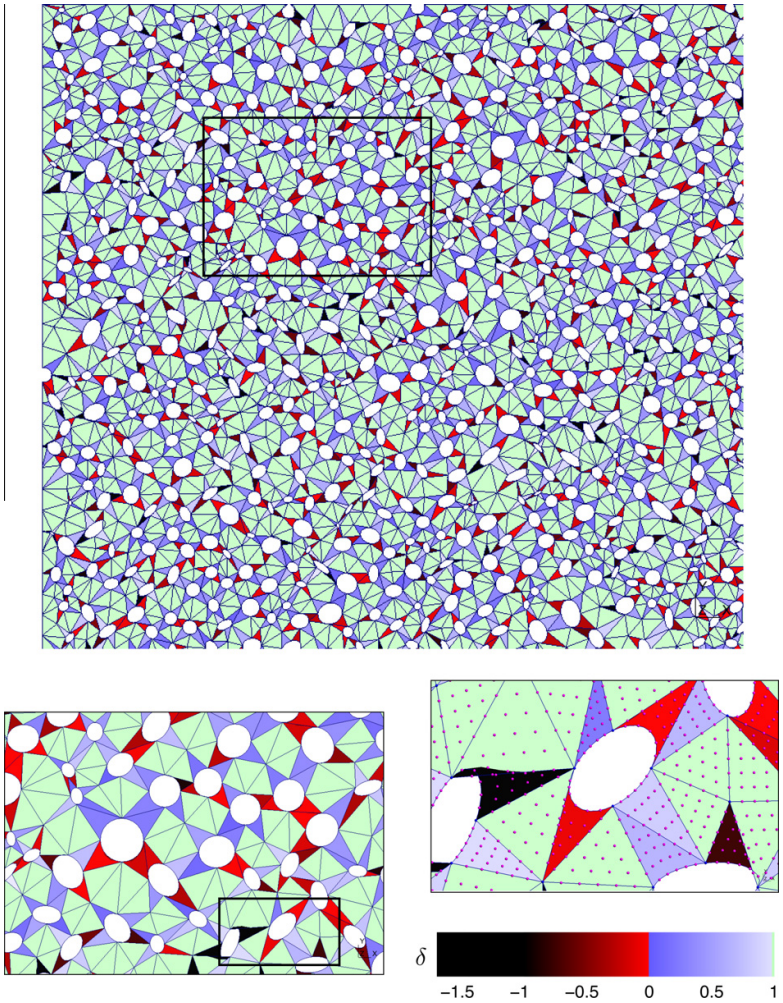


Fig. 4. Two-dimensional mesh with sixth order triangles; 47.5% of the elements are curved. The straight elements are in green and the curved ones are colored in function of the minimum of the distortion. The valid elements ($\delta_{\min} > 0$) are colored in blue. The invalid ones are colored in red if $\delta_{\min}$ is near 0 and in black if $\delta_{\min} < -1.5$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$$\mathcal{T}^{(p)}_{i,j,k}(\xi,\eta,\zeta) = \binom{p}{i}\binom{p-i}{j}\binom{p-i-j}{k}\ \xi^i\ \eta^j\ \zeta^k\ (1-\xi-\eta-\zeta)^{p-i-j-k} \quad (i+j+k \leqslant p),$$

$$\mathcal{P}^{(p)}_{i,j,k}(\xi,\eta,\zeta) = \mathcal{T}^{(p)}_{i,j}(\xi,\eta)\ \mathcal{B}^{(p)}_{k}(\zeta) \quad (i+j \leqslant p,\ k \leqslant p)$$

and

$$\mathcal{H}^{(p)}_{i,j,k}(\xi,\eta,\zeta) = \mathcal{B}^{(p)}_{i}(\xi)\ \mathcal{B}^{(p)}_{j}(\eta)\ \mathcal{B}^{(p)}_{k}(\zeta) \quad (i \leqslant p,\ j \leqslant p,\ k \leqslant p).$$

Matrices of change of coordinates can then be computed inline for every polynomial order, and bounds of Jacobian determinants computed accordingly. Table 1 summarizes the order of the Jacobian determinant and the number of coefficients in its expansion for all common element types. In all cases the subdivision scheme works exactly in the same way as for lines. Fig. 3 shows the first level of subdivision for a third-order triangle.

4.4. Implementation

As mentioned in Section 2, the bounds on the Jacobian determinant can be used to either make the distinction between valid and invalid elements with respect to a condition on $J_{\min}$, or to measure the quality of the elements by systematically computing $J_{\min}$ and $J_{\max}$ with a prescribed accuracy.

In both cases the same operations are executed on each element. First, the Jacobian determinant is sampled on a determined number of points N_s, equal to the dimension of the Jacobian determinant space, and so to the number of Bézier functions. Second, Bézier values are computed. Then adaptive subdivision is executed if necessary. Algorithms 2 and 3 show in pseudo-code the algorithm used to determine whether the Jacobian determinant of the element is everywhere positive or not.

Algorithm 2. Check if an element is valid or invalid

Input: a pointer to an element.
Output: *true* if the element is valid, *false* if the element is invalid

```
set sampling points P_i, i = 1,...,N_s;
compute Jacobian determinants J_i at points P_i;
for i = 1 to N_s do
    if J_i <= 0 then return false;
compute Bézier coefficients b_i, i = 1,...,N_s using (18);
i = 1;
while i ≤ N_s and b_i > 0 do
    i = i + 1;
if i > N_s then return true;
call algorithm 3 with b_i as arguments and return output;
```

Algorithm 3. Compute the control values of the subdivisions

Input: Bézier coefficients b_i, $i = 1,\ldots,N_s$
Output: *true* if the Jacobian determinant on the domain is everywhere positive, *false* if not

```
compute new Bézier coefficients b_i^[q], q = 1,...,Q as in equation (20);
for q = 1 to Q do
    for i = 1 to K_f do
        if b_i^[q] <= 0 then return false;
for q = 1 to Q do
    i = 1;
    while i ≤ N_s and b_i^[q] > 0 do
        i = i + 1;
    if i ≤ N_s then
        call algorithm 3 with b_i^[q] as arguments and store output;
        if output = false then return false;
return true;
```

Algorithm 3 could be further improved by optimzing the loop on line 5, by first selecting q for which we have the best chance to have a negative Jacobian determinant (line 4, algo 3). In practice this improvement is not significant since the only case for which we can save calculation is for invalid elements—and the proportion of them which require subdivision in order to be detected is usually small (about 3% for the mesh depicted in Fig. 4). Note that we may also want to find, for example, all the elements for which the Jacobian determinant is somewhere smaller than 20% of its average. We then just have to compute this average and replace the related lines (4 and 7 for Algorithm 2).

Another possible improvement is to relax the condition of rejection. We could accept elements for which all control values are positive but reject an element as soon as we find a Jacobian determinant smaller than a defined percent of the average Jacobian determinant. The computational gain can be significant, since elements that were classified as good and which needed a lot of subdivisions (and have a Jacobian determinant close to zero) will be instead rapidly be detected as invalid.

More interestingly, the computation of sampled Jacobian determinants and the computation of Bézier control values in Algorithm 2 can easily be executed for a whole groups of elements at the same time. This allows to use efficient BLAS 3 (matrix–matrix product) functions, which significantly speeds up the computations.

The algorithm used for all the tests in the next section is implemented in the open source mesh generator Gmsh [4] as the `AnalyseCurvedMesh` plugin.

5. Numerical results

We start by comparing the new adaptive computation of bounds on Jacobian determinants with the brute-force sampling of the Jacobian determinant for the detection of invalid high-order triangles.

The points at which we sample the Jacobian determinant for the brute-force method are taken as the nodes of an element of order k, for $k = 1, \ldots, 65$, leading to a number of sampling points comprised between 3 and 2211. In order to make the comparison as fair as possible, we have implemented the brute-force computation as efficiently as possible, i.e., for k ($> n$) sufficiently large we sample the Jacobian determinant on the points computed for an element at order n (the order

Table 2
On the left, number of curved elements detected as valid or invalid at each stage of the adaptive algorithm; at the first stage, 8309 elements can be classified as invalid due to a negative value of at least one of the 66 sampling points. Then Bézier coefficients are computed and 29,715 elements are classified as valid because those coefficients are positive. The 1224 (3.14%) remaining elements need to be subdivided adaptively. On the right, computation time; most of the time is spent on sampling the Jacobian determinant and computing the first Bézier coefficients.

	Curved element classification			# Elements analysed at given stage	CPU time for given stage[s]
	Valid elements	Invalid elements	Undertermined elements		
First stage	29,715	8,039	1,224	38,978	1.865
1 subdiv.	+787	+0	437	1,224	1.16e−1
2 subdiv.	+285	+17	135	437	8.40e−2
3 subdiv.	+56	+15	64	135	4.02e−2
4 subdiv.	+16	+22	26	64	2.40e−2
5 subdiv.	+5	+15	6	26	1.10e−2
6 subdiv.	+1	+2	3	6	4.34e−3
7 subdiv.	+1	+2	0	3	1.47e−3
Subtotal	30,866	8,112			2.146
Total	38,978				

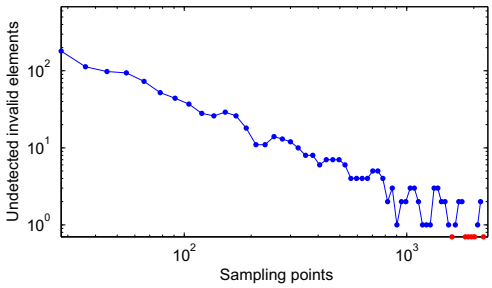


Fig. 5. Number of undetected invalid elements using brute-force sampling of the Jacobian determinant. The five red data points correspond to the correct result, i.e., when no invalid triangle is left undetected. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

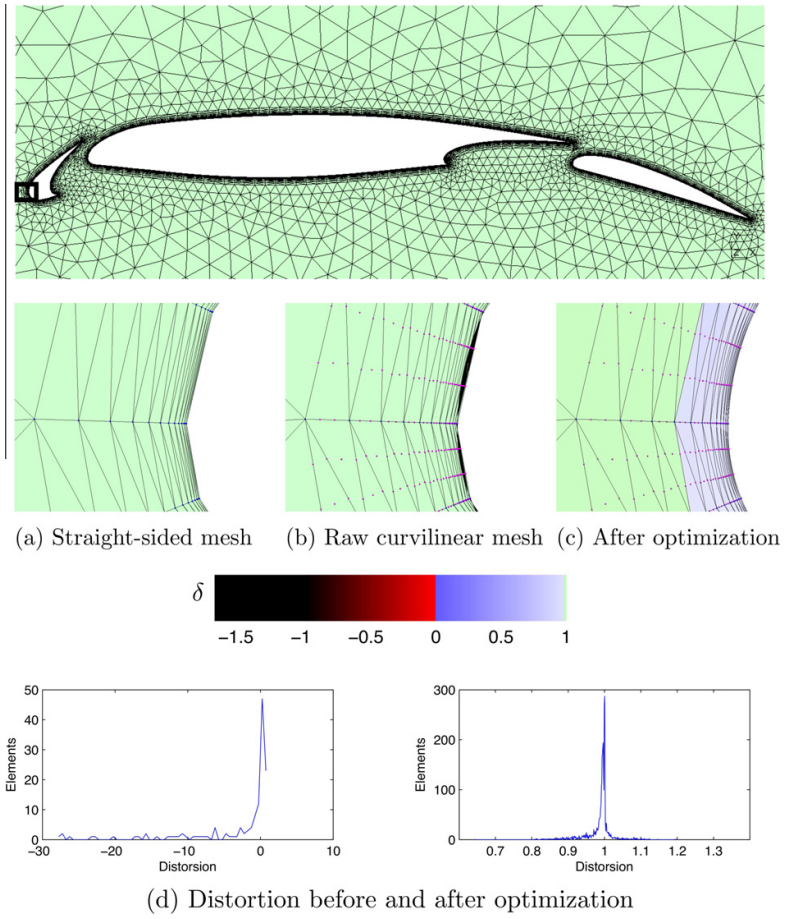


Fig. 6. Distortion of the curvilinear mesh of a wing (p3 triangles) before and after optimization.

of the Jacobian determinant) and then compute the desired Jacobian determinant values by a matrix–vector product, just like in our own adaptive method.

We consider the two-dimensional microstructure with oval holes depicted in Fig. 4, meshed with 82,009 sixth-order triangles. In this mesh 38,978 triangles are curved, and 8112 are invalid. The new algorithm successfully detects all the 8112 invalid elements in 2.146 s on a 2.4 GHz Core 2 Duo laptop computer.[1] Some elements needed as much as 7 levels of subdivisions in order to be classified: see Table 2. The brute-force approach required 1,596 sample points per triangle ($k = 55$) in order to detect all the invalid elements, and took more than 7 times longer. But far worse, increasing the number of sampling points beyond 1,596 can actually lead to a decreased accuracy of the prediction, as shown in Fig. 5.

Let us now examine the use of the adaptive bounds in the curvilinear meshing algorithm. We consider the boundary layer mesh of the B-Spline CAD model of a tri-part wing: see Fig. 6. The cubic triangular mesh is generated as follows. We first generate a straight-sided mesh (Fig. 6(a)). Then, every mesh edge that is classified on a model edge is curved by snapping their center vertices on the model edge. High order nodes are then inserted on every edge and in the middle of every face (Fig. 6(b)). This simple procedure does not guarantee that the final mesh is valid. In our case, 66 elements are invalid. Then, an optimization is applied globally (Fig. 6(c)). The final curvilinear mesh contains about 31% of curved elements. During the meshing process, the adaptive bound computation allowed to detect all invalid elements (the worst minimum of the distortion that was observed was $\delta_{\min} = -27.72$). After optimization, the final mesh is composed of elements that have $\delta_{\min} > 0.64$.

[1] Note that for completeness the algorithm also analyzes straight-sided elements, which is unnecessary in practice.

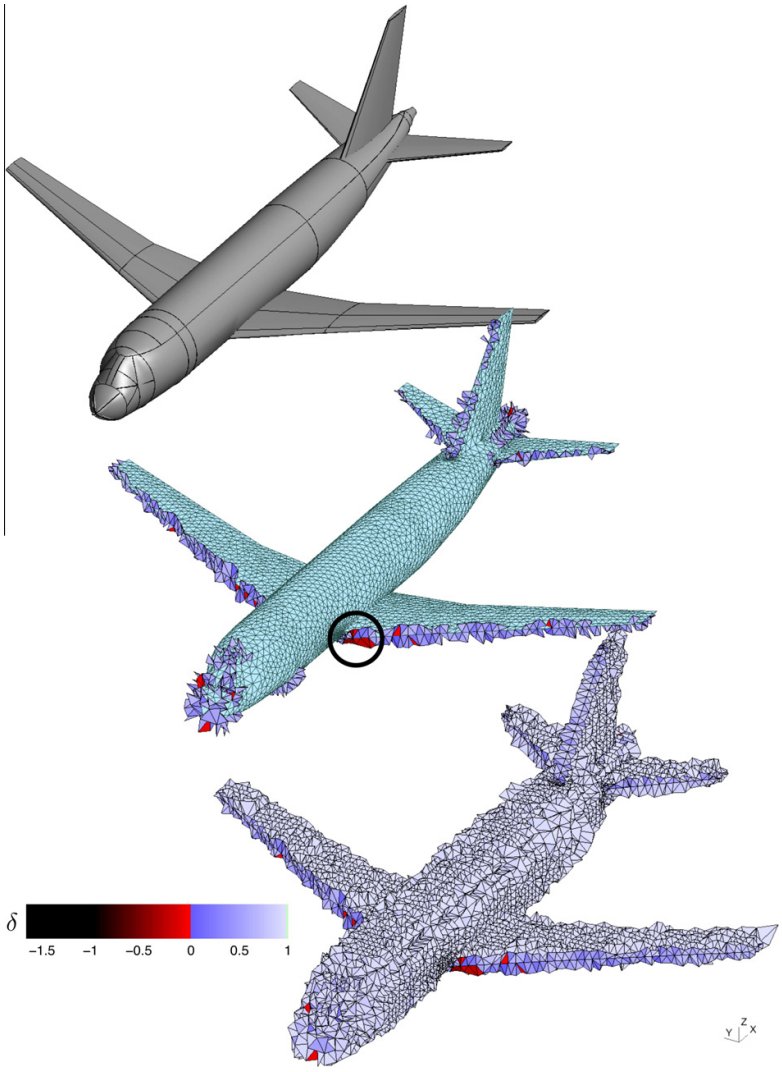


Fig. 7. The geometry of a A319 plane is meshed with p3 tetrahedron without executing optimization. The curved elements are all shown in the figure at the bottom. For the figure at the center, only elements for which $\delta \leqslant 0.7$ are shown. On 168,884 elements, 24,691 are curved and 76 are invalid.

Finally, a 3D mesh is considered. The CAD model of an A319 plane is meshed with 168,884 $p3$ tetrahedra (see Fig. 7 and 8). Without executing any optimization, 76 elements are invalid. The new algorithm detects them in 9.88 s on a 2.4 GHz Core 2 Duo laptop computer. The worst elements in term of their Jacobian determinant are located around leading edges, where the curvature is the most important. 999 elements have $\delta_{\min} \leqslant .7$ and the worst distortion is $\delta_{\min} = -7.74$.

6. Conclusions and perspectives

In this paper we presented a way to compute accurate bounds on Jacobian determinants of curvilinear finite elements, based on the efficient expansion of these Jacobian determinants in terms of Bézier functions.

The overall idea can be summarized as follows:

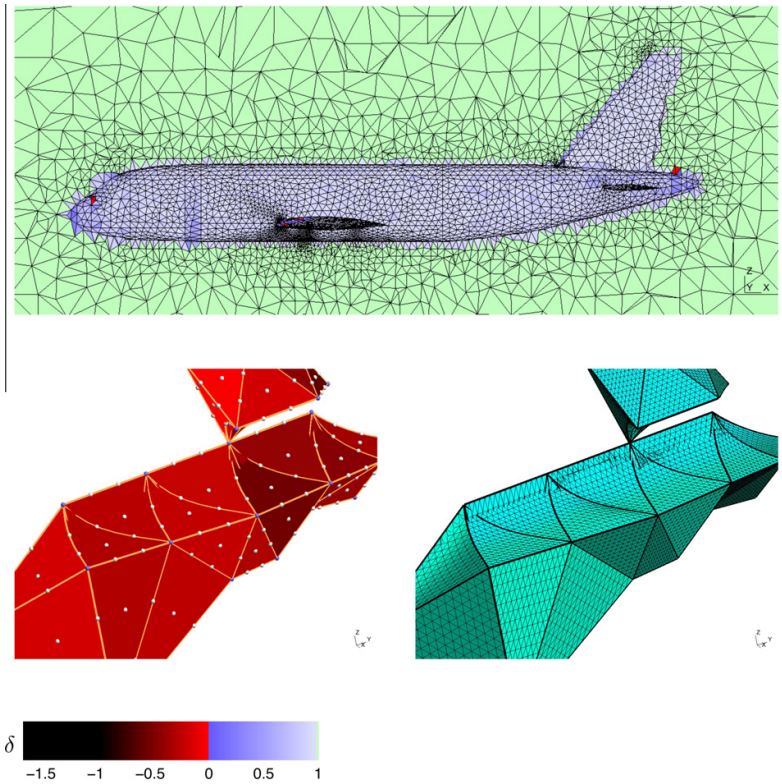


Fig. 8. The top figure shows a cut of the mesh of the A319 plane (figure 7). The two bottom figures show same invalid elements. On the left, elements are colored in function of their distortion. Here, we can clearly see that some faces intersect each other. In some cases, the Jacobian determinant is negative only inside the element so as it is not possible to see it visually.

1. The Jacobian determinant of a polynomial element is also a polynomial (of higher order).
2. The Bézier polynomial basis satisfies the convex hull property, which means that any polynomial expressed in this basis is bounded by the values at the control nodes.
3. By a change of polynomial basis (e.g. from Lagrange to Bézier), one naturally gets the bounds of the Jacobian determinant.
4. If the bounds are not accurate enough, one can subdivide the element, and once again, with a change of polynomial basis, obtain a more accurate bound in each sub-element (and so on and so forth until enough precision is reached).

The proposed algorithm can either be used to determine the validity or invalidity of curved elements, or provide an efficient way to measure their distortion. Triangles, quadrangles, tetrahedra, prisms and hexahedra can be analyzed using the same algorithm, which is available in the open source mesh generator Gmsh. Numerical tests show that the method is robust, and a user-defined error tolerance permits to adjust the accuracy vs. computational time ratio.

Perspectives for future work are numerous. We are currently investigating two related areas: first, we are working on a generalization of the bounds presented in this paper to the case of surface meshes embedded in 3D (curved surfaces). Second, we are investigating the use of various optimization strategies to generate meshes with provably good qualities.

Acknowledgements

This research Project was funded in part by the Walloon Region under WIST 3 Grant 1017074 (DOMHEX).

Authors gratefully thank E. Bechet from the University of Liège and K. Hillewaert from Cenaero for insightful discussions about Bézier functions and curvilinear mesh generation. Authors also thank V. D. Nguyen for providing the microstructure geometry used in Fig. 4.

References

[1] S. Dey, R.M. O'Bara, M.S. Shephard, Curvilinear mesh generation in 3D, Computer Aided Geometric Design 33 (2001) 199–209.
[2] M.S. Shephard, J.E. Flaherty, K.E. Jansen, X. Li, X. Luo, N. Chevaugeon, J.-F. Remacle, M.W. Beall, R.M. O'Bara, Adaptive mesh generation for curved domains, Applied Numerical Mathematics 52 (2005) 251–271.
[3] S.J. Sherwin, J. Peiró, Mesh generation in curvilinear domains using high-order elements, International Journal for Numerical Methods in Engineering 53 (2002) 207–223.
[4] C. Geuzaine, J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, International Journal for Numerical Methods in Engineering 79 (11) (2009) 1309–1331.
[5] X. Roca, A. Gargallo-Peiró, J. Sarrate, Defining quality measures for high-order planar triangles and curved mesh generation, in: Proceedings of the 20th International Meshing RoundTable, 2012, pp. 365–383.
[6] T. Hughes, The Finite Element Method, Dover, 2003.
[7] I. Babuška, B. Szabó, R.L. Actis, Hierarchic models for laminated composites, International Journal for Numerical Methods in Engineering 33 (1992) 503–535.
[8] G.E. Farin, Curves and Surfaces for CAGD: A Practicle Guide, Morgan-Kaufmann, 2002.
[9] J.M. Lane, R.F. Riesenfeld, A theoretical development for the computer generation and display of piecewise polynomial surfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 2 (1) (1980) 35–46.
[10] E. Cohen, L.L. Schumacker, Rates of convergence of control polygons, Computer Aided Geometric Design 2 (1985) 229–235.

APPENDIX D

Paper IV: Geometrical validity of high-order pyramidal finite elements

Journal of Computational Physics 299 (2015) 124–129

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Short note

Geometrical validity of curvilinear pyramidal finite elements

A. Johnen, C. Geuzaine *

Université de Liège, Department of Electrical Engineering and Computer Science, Montefiore Institute B28, Grande Traverse 10, 4000 Liège, Belgium

ARTICLE INFO

Article history:
Received 26 November 2014
Accepted 26 June 2015
Available online 3 July 2015

Keywords:
Finite element method
High-order methods
Mesh generation
Bézier functions

1. Introduction

A method to efficiently determine the geometrical validity of curvilinear finite elements of any order was recently proposed in [1]. The method is based on the adaptive expansion of the Jacobian determinant in a polynomial basis built using Bézier functions, that has both properties of boundedness and positivity. While this technique can be applied to all usual finite elements (triangles, quadrangles, tetrahedra, hexahedra and prisms), it cannot readily be applied to pyramids, due to non-polynomial nature of pyramidal finite element spaces.

In this short paper, we extend the results from [1] to pyramidal elements, by making use of the high-order nodal pyramidal finite element proposed by Bergot et al. [2], which exhibits optimal convergence properties in H^1-norm.

The paper is organized as follows. We begin by briefly recalling the pyramidal finite element space in Section 2, before constructing the function space of the Jacobian determinant in Section 3. Section 4 then introduces a generalized Bézier function basis, which can be used to obtain adaptive bounds on the pyramidal Jacobian determinant. Numerical results showing the sharpness of the estimates are given in Section 5.

2. Pyramidal finite element space

Let (ξ, η, ζ) denote the coordinates of the reference space and let $\mathcal{P}^r$ denote the pyramidal finite element space at order $r > 0$ defined in [2]. The space $\mathcal{P}^r$ can be expressed as the union of the classical tetrahedral finite space and the product of the triangular finite element space with powers of the non-affine term $\frac{\xi\eta}{1-\zeta}$:

$$\mathcal{P}^r := \left\{ \xi^i \eta^j \zeta^k \;\middle|\; i+j+k \le r \right\} \bigcup \left\{ \xi^i \eta^j \left(\frac{\xi\eta}{1-\zeta} \right)^{r-l} \;\middle|\; i+j \le l,\ l \le r-1 \right\}. \tag{1}$$

* Corresponding author.
E-mail addresses: a.johnen@ulg.ac.be (A. Johnen), cgeuzaine@ulg.ac.be (C. Geuzaine).

http://dx.doi.org/10.1016/j.jcp.2015.06.033

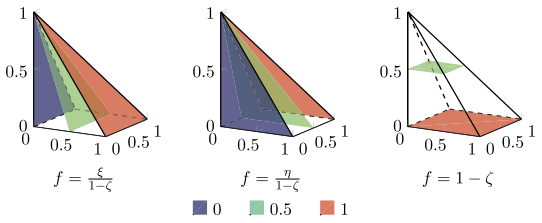


Fig. 1. Visualization of the three subfunctions that generate the pyramidal nodal space.

In the previous expression all the indices (i, j, k, l) are assumed to be integers greater than or equal to 0. (The same convention is used for all indices throughout the paper.) The previous definition can be rewritten in a more convenient form:

$$\mathcal{P}'^r := \left\{ \left(\frac{\xi}{1-\zeta}\right)^i \left(\frac{\eta}{1-\zeta}\right)^j (1-\zeta)^k \;\middle|\; i, j \le k,\ k \le r \right\} = \mathcal{P}^r. \tag{2}$$

Proof. We can separate the space $\mathcal{P}'^r$ into two subspaces, one for which $k \ge i + j$ and the other for which $k < i + j$. We have

$$\mathcal{P}'^r\Big|_{k \ge i+j} = \left\{ \xi^i \eta^j (1-\zeta)^K \;\middle|\; i + j + K \le r \right\},$$

which is the tetrahedral space. Let us rewrite the second contribution in terms of $\xi^I \eta^J \left(\frac{\xi\eta}{1-\zeta}\right)^K$. We then have $i = I + K$, $j = J + K$ and $k - i - j = -K$, which implies:

$$\mathcal{P}'^r\Big|_{k < i+j} = \left\{ \xi^I \eta^J \left(\frac{\xi\eta}{1-\zeta}\right)^K \;\middle|\; I + J + K \le r,\ 1 \le K \right\}.$$

By substituting $r - l$ for K, we obtain the non-affine part of Bergot's pyramidal space. And thus eventually $\mathcal{P}'^r = \mathcal{P}^r$. □

In addition to being more convenient for the developments of Section 3, the form (2) offers the advantage of showing that functions of the pyramidal space are generated by the product of integer powers of three elementary subfunctions: $\frac{\xi}{1-\zeta}$, $\frac{\eta}{1-\zeta}$ and $1 - \zeta$ (see Fig. 1). The first, $\frac{\xi}{1-\zeta}$, is equal to one on face $\xi = 1 - \zeta$ and equal to zero on face $\xi = 0$. The second, $\frac{\eta}{1-\zeta}$, is equal to one on face $\eta = 1 - \zeta$ and zero on face $\eta = 0$. The third, $1 - \zeta$, is equal to 1 on face $\zeta = 0$ and equal to 0 on the top corner. It is thus similar to what we have for tetrahedra or hexahedra, whose finite element spaces are spanned by the product of integer powers of the subfunctions ξ, η and ζ.

It is easy to see in form (1) that the basis functions are continuous since $\frac{\xi\eta}{1-\zeta}$ is well-defined at the top corner $(0, 0, 1)$. In the second form (2), the functions $\left(\frac{\xi}{1-\zeta}\right)^i$ and $\left(\frac{\eta}{1-\zeta}\right)^j$ are not well-defined but their product with $(1 - \zeta)^k$ is well-defined since $k \ge \max(i, j)$.

The pyramidal finite element is characterized by the mapping between a reference pyramid and the actual pyramid in the mesh. To be valid, this mapping should be bijective, which implies that the Jacobian determinant should be positive everywhere inside the domain of definition [1]. This is why, in the following two sections, we first construct the function space of the Jacobian determinant and then present its Bézier expansion.

3. Pyramidal Jacobian determinant space

Let $\mathcal{J}^r$ denote the Jacobian determinant space. We have by definition $\mathcal{J}^r = \mathcal{P}^r_{,\xi} \times \mathcal{P}^r_{,\eta} \times \mathcal{P}^r_{,\zeta}$, where $\mathcal{P}^r_{,\bullet}$ is the space obtained by differentiating all the elements of space $\mathcal{P}^r$ with respect to $\bullet$. From (2), we obtain:

$$\mathcal{P}^r_{,\xi} = \left\{ \left(\frac{\xi}{1-\zeta}\right)^{i_1} \left(\frac{\eta}{1-\zeta}\right)^{j_1} (1-\zeta)^{k_1} \;\middle|\; i_1 \le k_1,\ j_1 \le k_1 + 1,\ k_1 \le r - 1 \right\}$$

$$\mathcal{P}^r_{,\eta} = \left\{ \left(\frac{\xi}{1-\zeta}\right)^{i_2} \left(\frac{\eta}{1-\zeta}\right)^{j_2} (1-\zeta)^{k_2} \;\middle|\; i_2 \le k_2 + 1,\ j_2 \le k_2,\ k_2 \le r - 1 \right\}$$

$$\mathcal{P}^r_{,\zeta} \subset \left\{ \left(\frac{\xi}{1-\zeta}\right)^{i_3} \left(\frac{\eta}{1-\zeta}\right)^{j_3} (1-\zeta)^{k_3} \;\middle|\; i_3, j_3 \le k_3 + 1,\ k_3 \le r - 1 \right\}.$$

The inclusion in the last expression arises from a simplification: we do not discard the case in (2) corresponding to $k - i - j = 0$, which should be discarded when differentiating with respect to ζ. Considering the real space of $\mathcal{P}^r_{,\zeta}$ would only complicate further developments, and not provide any other advantages.

The product of the three spaces leads to the following expression for the Jacobian determinant space:

$$\mathcal{J}^r \subset \left\{ \left(\frac{\xi}{1-\zeta}\right)^I \left(\frac{\eta}{1-\zeta}\right)^J (1-\zeta)^K \;\middle|\; I, J \leq K+2, \;\; K \leq 3r-3 \right\}, \tag{3}$$

which implies that $\mathcal{J}^r$ is a subset of $\mathcal{P}^{3r-3} \times \left\{ \left(\frac{\xi}{1-\zeta}\right)^i \left(\frac{\eta}{1-\zeta}\right)^j \middle| i, j \leq 2 \right\}$, whose dimension is $\sum_{k=0}^{3r-3}(k+3)^2 = r/2 \times (3r+1)(6r+1) - 5$. We see that, while for other element types the Jacobian determinant space is contained in their finite element space of a higher order [1], for pyramids, this is not the case.

The pyramidal Jacobian determinant is not well-defined at the top corner: K can be smaller than the maximum of I and J in which case the term $(1-\zeta)^K$ cannot fully compensate the two other terms. As a consequence, one should never sample the Jacobian determinant at the top corner of the pyramid.

4. Bézier basis for the pyramidal Jacobian determinant

While the use of Bézier interpolation to parametrize curves and surfaces is very common in computer graphics, it is less so to expand general functions. One property of Bézier expansion that is useful for our problem is that the interpolant is located inside the convex hull of the control values. This property allows to provide bounds on the interpolant. All positive basis functions that sum up to 1 have this property but, intuitively, the Bézier basis is the one for which the size of the convex hull is the smallest (thus, for which the bounds are the sharpest). Another desired property of Bézier expansion is that it can be recursively "subdivided" [1] which allows to sharpen the bounds.

Polynomial Bézier bases are based on the Bernstein polynomials. At order n, the $n+1$ Bernstein polynomials are defined as

$$B_k^{(n)}(\lambda) := \binom{n}{k} \lambda^k (1-\lambda)^{n-k} \qquad (k = 0, \ldots, n),$$

where $\binom{n}{k} = \frac{n!}{n!(n-k)!}$ is the binomial coefficient. They sum up to 1 and they are positive on the domain $[0, 1]$. In order to compute bounds in the non-polynomial pyramidal Jacobian determinant space, we will search for a basis that can be written as product of a generalization of Bernstein polynomials.

4.1. Generalized Bézier basis for pyramids

Let $\Omega_{\text{ref}} \subset \mathbb{R}$ denote the uncentered pyramid, for which $(\frac{\xi}{1-\zeta}, \frac{\eta}{1-\zeta}, 1-\zeta) \in [0,1]^3$. (As usual for Bézier interpolation we will define the Jacobian determinant basis functions on this uncentered pyramid Ω_{ref} instead of the centered pyramid that is often used in finite element methods.) From (3), we easily identify the Jacobian determinant basis written with generalized Bernstein functions:

$$J^r_{i,j,k}(\xi, \eta, \zeta) := B_i^{(k+2)}\left(\frac{\xi}{1-\zeta}\right) \; B_j^{(k+2)}\left(\frac{\eta}{1-\zeta}\right) \; B_k^{(3r-3)}(1-\zeta), \qquad (\xi, \eta, \zeta) \in \Omega_{\text{ref}}. \tag{4}$$

Like for hexahedra and prisms, the Jacobian determinant of the first-order pyramid is not constant. However it is a function of only $\frac{\xi}{1-\zeta}$ and $\frac{\eta}{1-\zeta}$. This means that sampling of the Jacobian determinant can be done on the $\zeta = 0$ plane, and that recursive subdivision works in the same way as for the quadrangle element [1].

For high-order pyramids, definition (4) is relevant if subdivision is not required (e.g. for optimization). But as explained in the following subsection, recursive subdivision with respect to the ζ-axis does not hold, which motivates the definition of an enriched basis.

4.2. Enriched generalized Bézier basis for pyramids

Let Ω_{bot} denotes the bottom subdomain obtained when cutting the reference pyramid by the plane $\zeta = 1/2$. We note $\boldsymbol{M}_{\text{bot}} : \Omega_{\text{bot}} \to \Omega_{\text{ref}}$ the mapping between the bottom subdomain and the reference pyramid. We have:

$$\boldsymbol{M}_{\text{bot}} : \begin{cases} \xi' \mapsto \xi = \xi' \frac{1-2\zeta'}{1-\zeta'} \\ \eta' \mapsto \eta = \eta' \frac{1-2\zeta'}{1-\zeta'} \\ \zeta' \mapsto \zeta = 2\zeta' \end{cases}.$$

Recursive subdivision is possible for the bottom if the Jacobian determinant can be expanded into the basis whose functions are $S^r_{i,j,k} := J^r_{i,j,k} \circ \boldsymbol{M}_{\text{bot}}$. Those functions are defined on Ω_{bot} and have properties of positivity and partition of unity. Their expression is:

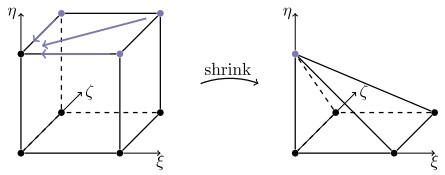


Fig. 2. The pyramid can be seen as a shrunk cube with the transformation $\xi \mapsto \frac{\xi}{1-\zeta}$ and $\eta \mapsto \frac{\eta}{1-\zeta}$.

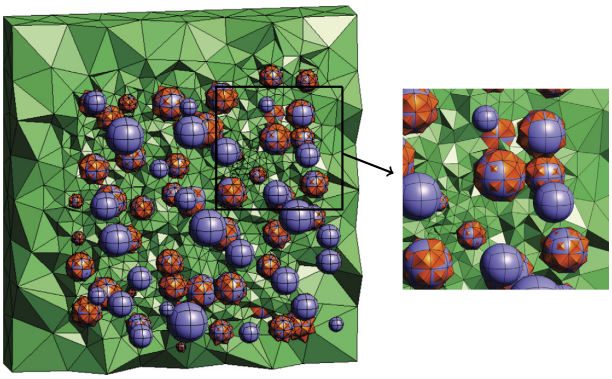


Fig. 3. Three-dimensional mesh with second-order elements. The geometry consists of a cube with spherical holes. Pyramids (in orange) make the transition from the hexahedra (in blue) that fill the holes to the tetrahedra (in green) that fill the rest of the volume. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$S^r_{i,j,k}(\xi,\eta,\zeta) = B_i^{(k+2)}\left(\frac{\xi}{1-\zeta}\right)\ B_j^{(k+2)}\left(\frac{\eta}{1-\zeta}\right)\ B_k^{(3r-3)}(1-2\zeta)\,, \qquad (\xi,\eta,\zeta)\in\Omega_{\text{bot}}, \tag{5}$$

but it can be shown that they do not span the Jacobian determinant space due to the dependence on k of the two first Bernstein functions. We therefore define the enriched Jacobian determinant basis functions by removing this dependence:

$$E^r_{i,j,k}(\xi,\eta,\zeta) := B_i^{(3r-1)}\left(\frac{\xi}{1-\zeta}\right)\ B_j^{(3r-1)}\left(\frac{\eta}{1-\zeta}\right)\ B_k^{(3r-3)}(1-\zeta)\,, \qquad (\xi,\eta,\zeta)\in\Omega_{\text{ref}}. \tag{6}$$

These functions correspond to the ones one would obtain by considering a “shrinked” cube (Fig. 2). The corresponding basis can be recursively and adaptively subdivided.

As described in [1], fast computation of Bézier coefficients can be achieved by using a transformation matrix that computes control values from nodal values. The Jacobian determinant is sampled at the location of the nodes of a pyramid of order $3r-1$, excepted the node at the top and the four nodes directly below the top. Subdivision works in exactly the same way as for other element types, provided that for the first-order pyramid, subdivision is only necessary along the base of the pyramid.

5. Results

We present the results of our algorithm applied to a three-dimensional microstructure. The structure contains spherical holes that are meshed with second order hexahedra. In order to make the transition with the second-order tetrahedra that are used for the rest of the geometry, second-order pyramids have been generated, around those holes (see Fig. 3). We measure the minimum of distortion δ_{min}, *i.e.* the minimum of the determinant of the mapping between the straight-sided element and the curved element, as defined in [1]. The analyzed mesh is composed of 180,356 tetrahedra for which 31,696 are curved and 5809 curved pyramids.

We improved the algorithm presented in [1] in order to compute δ_{min} with a given input tolerance ε and detect the invalid elements at the same time. First, we compute the Bézier coefficients of the whole element. Then we enter in a loop:

1. Compute $\delta^{\text{sup}}_{\text{min}}$ and $\delta^{\text{inf}}_{\text{min}}$ (upper and lower bound on δ_{min}) as in [1]
2. If $\delta^{\text{sup}}_{\text{min}} - \delta^{\text{inf}}_{\text{min}} \le \varepsilon$ and $\delta^{\text{sup}}_{\text{min}}\,\delta^{\text{inf}}_{\text{min}} \ge 0$, then go to 4
3. Subdivide the (sub)domain that contains the smaller Bézier coefficient and go to 1
4. Return $\delta^{\text{inf}}_{\text{min}}$ (NB: the element is invalid if $\delta^{\text{inf}}_{\text{min}} \le 0$, else it is valid)

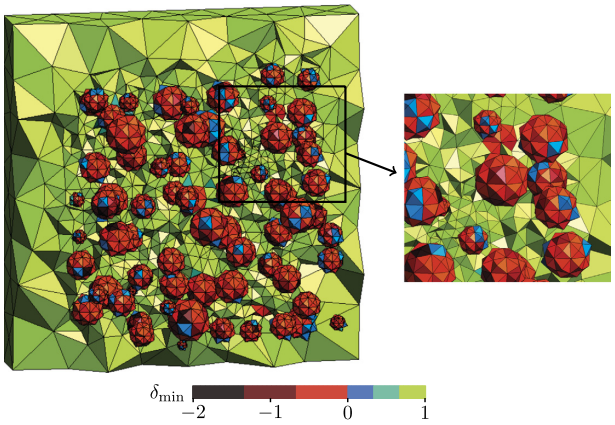


Fig. 4. Validity of the mesh. Valid elements are between green and blue and invalid elements are between red and black. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

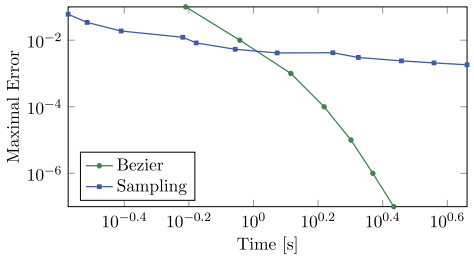


Fig. 5. Analysis of 186,165 second-order elements.

Fig. 4 presents the results on the mesh. Our algorithm successfully detects the 4989 invalid pyramids and the 82 invalid tetrahedra (elements in red to black).

Fig. 5 compares the computation time versus the maximal error of our algorithm and the brute-force sampling of the Jacobian determinant. For our algorithm, we measure the time taken to compute bounds with an input tolerance of $\varepsilon = 10^{-e}$, $e = \{1, \ldots, 7\}$. For the brute-force sampling, the Jacobian determinant is sampled at an increasing number of points. In the case of tetrahedral elements, the sampling points are chosen as the nodes of a tetrahedron of order k. For pyramidal elements, the points are taken as the nodes of a pyramid of order $k+1$ for which we remove the five top nodes, *i.e.*, the same way we sample the Jacobian determinant for our method. The order k is ranged from 1 to 12 which involves that the number of sampling points is comprised between 4 and 455 for tetrahedra and between 9 and 2352 for pyramids. We measured the computation time and the maximal (elementary) error between $\min_{\text{sampling}}(\delta)$ and the best approximation of $\delta_{\min}$ taken as the value computed by our algorithm at tolerance 10^{-7}. Tests have been performed on a Macbook Pro Retina, Mid 2012 @ 2.3 GHz.

The brute-force sampling needs more time than our algorithm to reach a maximal error smaller than 4×10^{-3}. But worse, similarly to the results reported in [1] for other element types, the brute-force algorithm is not able to find all invalid pyramids for $k = \{1, \ldots, 12\}$, the maximum number of invalid pyramids found being 4971 (instead of 4989) at $k = 12$.

6. Conclusion

In this paper we adapted the computation of accurate bounds on Jacobian determinants of curvilinear finite elements to the pyramidal case. The proposed algorithm can either be used to determine the validity or invalidity of curved pyramids, or to provide an efficient way to measure their distortion. The complete implementation of the algorithm is available in the open source mesh generator Gmsh [3] as the `AnalyseCurvedMesh` plugin. Ongoing research includes adaptation of the algorithm to the computation of accurate bounds on a quality measure of the elements based on the metric.

Acknowledgement

This research project was funded in part by the Walloon Region under WIST 3 grant 1017074 (DOMHEX).

References

[1] A. Johnen, J.-F. Remacle, C. Geuzaine, Geometrical validity of curvilinear finite elements, J. Comput. Phys. 233 (2013) 359–372.
[2] M. Bergot, G. Cohen, M. Duruflé, Higher-order finite elements for hybrid meshes using new nodal pyramidal elements, J. Sci. Comput. 42 (3) (2010) 345–381, http://dx.doi.org/10.1007/s10915-009-9334-9.
[3] C. Geuzaine, J. Remacle, Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities, Int. J. Numer. Methods Eng. 79 (11) (2009) 1309–1331.

APPENDIX E

Paper V: Computing the extrema of a shape quality measure for curvilinear finite elements (Draft)

Computing the Extrema of a Shape Quality Measure for Curvilinear Finite Elements (Draft)

A. Johnen[1], T. Toulorge[2] and C. Geuzaine[1]

December 17, 2015

1 Introduction

With recent developments in the field of high-order finite element methods [1], such as discontinuous Galerkin [2] or spectral [3, 4] methods, there is a renewed interest for high-order (curved) mesh generation.

The classical finite element method, a.k.a. the h-version, uses linear elements to discretize the geometry and a refinement of the mesh is performed in order to increase the accuracy of the solution. It has been established that the p-version of the finite element, for which the order of the functions is increased in order to improve the accuracy, may provide better convergence [5]. Eventually, "super-convergence" can be obtained by a mix of the two approaches [6]. There has thus been a frenzy in the 1980s to develop such methods, but developers encountered several difficulties which hampered their momentum [7]... and consequently most of the current industrial-grade and commercial finite element packages are still based on at most second-order meshes. One reason is that during the process of generating a high-order curvilinear mesh, invalid (tangled) elements are often created, and untangling those is not a trivial task. Recent methods have improved the robustness of the untangling procedure through optimization, albeit at a high computational cost [8]. For this reason, the tendency is to apply the technique only on small groups of elements in the neighborhood of the invalid elements to untangle. It is therefore crucial to be able to detect invalid and poor-quality curvilinear elements.

A finite element is defined by the position of its nodes through a mapping between a reference element and itself. Its validity can thus be assessed by verifying the positivity of the determinant of the Jacobian of this mapping. It was thought for years that the only way to determine with certainty the

validity of non-trivial elements would be by computing the Jacobian determinant at an infinite number of points [9, 10]. Recent developments based on Bézier curves showed that it is nothing of the sort. A first step has been taken in [11] where it is shown that it is possible to compute bounds on the Jacobian determinant of second order tetrahedra; those bounds are however not sharp. References [12, 13] provided a complete solution by developing an adaptive technique for efficiently computing the minimum and the maximum of the Jacobian determinant of any type and any order of elements, up to any prescribed tolerance. This method subsequently allows to guarantee the validity of any element.

Validity is one aspect that influences the accuracy of solution. Another aspect is the quality of the finite elements. A distinction can be made between geometric quality measures and Jacobian-based quality measures. Geometric quality measures have been used since the very early days of finite element modeling and are constructed from geometric characteristics such as the area/volume of the element, the length of the edges or the radii of the inscribed and circumscribed circles/spheres [14, 15]. These geometric quality measures are however not easily generalizable to curved elements—see e.g. [11] for the extension to quadratic tetrahedra. Jacobian-based measures are a more natural fit, since the Jacobian matrix is defined for every order and every type of element. A framework that allows the construction, classification, and evaluation of such measures defined on linear elements has been proposed in [16, 17]. It is important to understand that Jacobian-based measures are essentially pointwise (within the element). For the linear triangles and tetrahedra, this is not a problem since the Jacobian matrix is constant. For other elements, an element-wise measure has to be extracted from the pointwise measure. In the two reference above, it is proposed to compute the measure at the corners of linear quadrangles and hexahedra and to take the minimum or the harmonic or geometric average. Similarly, for quadratic triangles, the element-wise measure can be computed as the minimum, maximum or the p^{th} power-mean of the pointwise measure sampled at the six nodes of the elements, although it is shown that in some situations this constitutes a poor approximation of the true minimum, maximum or p^{th} power-mean [10].

Recent works have focused on defining a quality measure for curved simplicial finite elements of any order [18, 19]. The general approach is to consider an algebraic quality measure as proposed in [16], which constitutes the pointwise measure, and to compute the L^2-norm as the element-wise quality measure. This measure however cannot be used as a validity test. It is mentioned in [20] that this technique, although developed only for simplicial

elements, can be extended to non-simplicial elements.

In this paper, we propose to extend the method that efficiently computes the extrema of the Jacobian determinant [12, 13] to a Jacobian-based quality measure. Instead of computing the element-wise quality measure by simply taking a norm, we aim at finding the actual bounds of the pointwise measure.

The paper is organized as follows. In Section 2, we begin by recalling the Jacobian matrix of the different mappings of an element before establishing a Jacobian-based quality measure. In Section 3, we present the Bézier expansion, we recall the algorithm for computing bounds on the validity of the elements [12] and we give some properties of Bézier expansions that are useful for computing bounds on the quality measure. In Sections 4 and 5, we explain how to compute the bounds for respectively 2D and 3D elements. An improvement for 3D elements is then presented in Section6. Finally, results are presented in Section 7 and we conclude in Section 8.

2 A quality measure based on the metric eigenvalues

Let us consider a 2D or 3D mesh of order n, which consists of a set of curved physical elements that can either be triangles or quadrangles for a 2D mesh and tetrahedra, hexahedra, prisms or pyramids for a 3D mesh. Let d_m and d_s denote respectively the dimension of the mesh and the dimension of the physical space in which the mesh is embedded. The couple (d_m, d_s) is equal to $(3, 3)$ for a 3D mesh and for a 2D mesh it can be equal to $(2, 2)$ or $(2, 3)$, meaning that the 2D mesh can either live in the xy-plane or be embedded inside a non-planar 3D surface. Each physical element is defined geometrically through its nodes $\boldsymbol{n}_k \in \mathbb{R}^{d_s},\ k = 1, \dots, N$ and a set of Lagrange shape functions $L_k^n(\boldsymbol{\xi}) : \Omega_{\text{ref}} \subset \mathbb{R}^{d_m} \to \mathbb{R},\ k = 1, \dots, N$. These are polynomial functions that allow to map a reference unit element, whose domain of definition is Ω_{ref}, onto the physical one (see Fig. 1):

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{k=1}^{N} L_k^n(\boldsymbol{\xi})\, \boldsymbol{n}_k.$$

Since we also consider pyramids, saying that the shape functions are Lagrange functions and thus polynomials is an abuse of language. But it has be shown that computing bounds on the Jacobian determinant of pyramids using the technique described in [12] is done as if it was a polynomial [13]. For simplicity, we will consider that all the elements have polynomial shape functions.

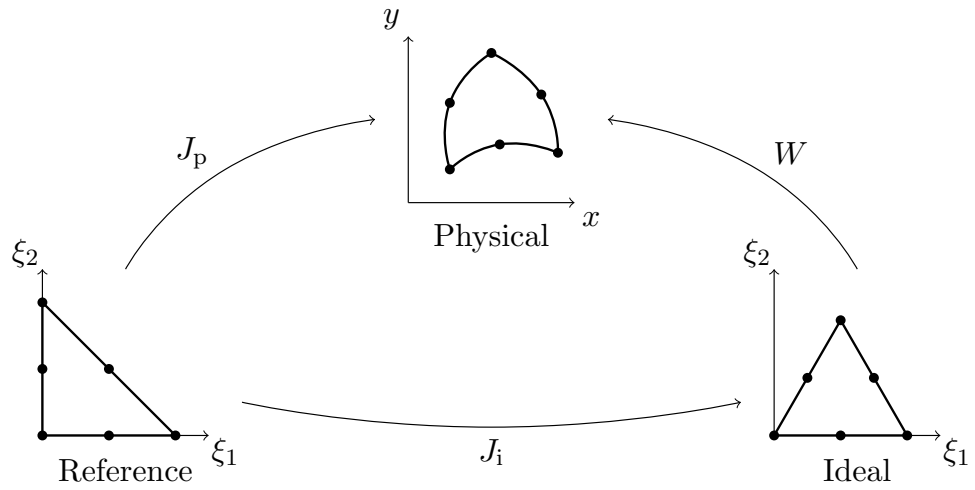


Figure 1: We consider three mappings, each of them is characterized by a Jacobian matrix: (1) J_p for the mapping between the reference and the physical element, (2) J_i for the mapping between the reference element and the ideal element and (3) W for the mapping between the ideal and the physical element. The latter is the one used to construct quality measures.

The Jacobian matrix of this mapping, denoted by J_p, is a matrix-valued function, and we have $J_\mathrm{p} : \Omega_\mathrm{ref} \to \mathbb{R}^{d_s \times d_m} : \boldsymbol{\xi} \mapsto J_\mathrm{p}(\boldsymbol{\xi})$. It is by definition the matrix of the first-order partial derivatives, *i.e.* $(J_\mathrm{p})_{ij} = \frac{\partial x_i}{\partial \xi_j}$ which are polynomials. J_p contains all the information of the element deformation with respect to the reference element and can naturally be used to construct a shape quality measure. However, as explained in [16], a quality measure should compare the physical element to an *ideal* element which defines the ideal shape, volume and/or orientation. The nature of the ideal elements is problem-dependent. For instance, the ideal triangle is usually the equilateral triangle but it can be an elongated triangle if the physics implies that those triangles give a better precision for the solution. The ideal element is thus either defined by the user or by the mesh generator. We denote the Jacobian matrix of the mapping between the reference and the ideal element, whose dimension is $d_m \times d_m$, by J_i.

The product $W := J_\mathrm{p}\, J_\mathrm{i}^{-1}$ gives the Jacobian matrix of the mapping between the ideal and the physical element and can be used in order to construct a quality measure. In this paper, we make the assumption that J_i is a constant matrix. The ideal elements we consider are therefore linear, have planar faces and, when appropriate, parallel opposite sides. For instance, our ideal quadrangle can be a parallelogram but not a trapeze. This implies

that the elements of W remain polynomials, which is a necessary condition for computing the bounds the way it is described in this paper.

Now, let us consider the following quantity defined on any element E:

$$\mu_{\text{pw}}(E, \boldsymbol{\xi}) = \frac{1}{||W||_2 \, ||W^{-1}||_2} \qquad \boldsymbol{\xi} \in \Omega_{\text{ref}} \,,$$

where $||\cdot||_2$ is the 2-norm of the matrix. As stated in Proposition 9.4 of [16], this corresponds to the distance of W to the set of singular matrices (matrices whose determinant is zero) and thus, μ_{pw} measures the pointwise distance to a locally degenerate element. It is thus a positive quantity which reaches the maximum value of 1 when the element is locally the furthest away possible from being degenerate. It follows also from Propositions 8.6 and 9.3 of the same paper that this quantity is a shape quality measure. In order to define an element-wise measure μ, we take the minimum of μ_{pw}:

$$\mu(E) = \min_{\boldsymbol{\xi} \in \Omega_{\text{ref}}} \frac{1}{||W||_2 \, ||W^{-1}||_2}.$$

Geometric interpretation The product $W^T W$ defines a squared matrix called the metric tensor which we denote by $\mathcal{M}$. A known result of matrix algebra is that $||W||_2$ is equal to the square root of the maximum eigenvalue of $\mathcal{M}$ [21]. Let λ_- and λ_+ be respectively the minimum and the maximum eigenvalue of $\mathcal{M}$. Those eigenvalues being positive, it is easy to demonstrate that

$$\frac{1}{||W||_2 \, ||W^{-1}||_2} = \sqrt{\frac{\lambda_-}{\lambda_+}}.$$

We denote the ratio $\frac{\lambda_-}{\lambda_+}$ by R. Thus the element-wise measure we want to compute is $\mu(E) = \min_{\boldsymbol{\xi} \in \Omega_{\text{ref}}} \sqrt{R(\boldsymbol{\xi})}$. In the geometrical point of view, it is well-known that the square root of the minimum (resp. maximum) eigenvalue of $\mathcal{M}$ corresponds to the smallest (resp. largest) length $||W\boldsymbol{u}||$ that can be obtained with a unit vector $\boldsymbol{u}$ (see, *e.g.*, [8]). It follows that $R(\boldsymbol{\xi}) = 1$, which corresponds to $\lambda_- = \lambda_+$, implies that the element is locally isotropic; and the more $R(\boldsymbol{\xi})$ approaches 0, the more the element is locally anisotropic.

Before introducing the Bézier expansion, we establish an result that will be helpful in this paper: when the Jacobian matrix is square, the product of the eigenvalues of $\mathcal{M}$ is equal to the square of the Jacobian determinant. Indeed, we have $\prod_{i=0}^{d_m} \lambda_i = \det(\mathcal{M}) = \det(W^T W) = \det(W)^2$.

3 Bézier expansion

Let $B^\nu_{\boldsymbol{i}}(\boldsymbol{\xi}) : \Omega_{\text{ref}} \subset \mathbb{R}^{d_m} \to \mathbb{R},\ \boldsymbol{i} \in \mathcal{I}^\nu$ denote a Bézier function of order ν. We use the multi-index notation: $\boldsymbol{i} = (i_1, \ldots, i_{d_m})$ is an ordered tuple of d_m indices. $\mathcal{I}^\nu$, which depends on the type of element, is the index set of the Bézier functions and $\{B^\nu_{\boldsymbol{i}}\}_{\boldsymbol{i}\in\mathcal{I}^\nu}$ defines the *Bézier basis*. The analytical expression of Bézier functions for linear, triangular, quadrangular, tetrahedral, hexahedral and prismatic elements are given in [12] and the expression for pyramidal elements is given in [13]. Let $f_{\boldsymbol{i}}$ denote the coefficients of the expansion. They are also known as the *control values*. For any polynomial function $f : \Omega_{\text{ref}} \subset \mathbb{R}^{d_m} \to \mathbb{R}$ of order at most ν, one can compute the control values such that we have the following equality:

$$f(\boldsymbol{\xi}) = \sum_{\boldsymbol{i}\,\in\,\mathcal{I}^\nu} f_{\boldsymbol{i}}\, B^\nu_{\boldsymbol{i}}(\boldsymbol{\xi}),$$

where the right member is the *Bézier expansion* of the function f.

The Bézier functions are positive and sum up to one which implies the well-known *convex hull* property. In our case, the convex hull property says that f is bounded by the extrema of the control values, *i.e.* $\min_{\boldsymbol{i}} f_{\boldsymbol{i}} \leq f(\boldsymbol{\xi}) \leq \max_{\boldsymbol{i}} f_{\boldsymbol{i}}$. In addition to that, there are control values that are actual values of the expanded function. Those control values are "situated" on the corners of the element [1] and we refer to their index set by $\mathcal{I}^\nu_c$. As a consequence, the control values allow to bound the two extrema of the function from below and above. For example, for the minimum we have: $\min_{\boldsymbol{i}} f_{\boldsymbol{i}} \leq f_{\min} \leq \min_{\boldsymbol{i}\in\mathcal{I}^\nu_c} f_{\boldsymbol{i}}$.

It has been shown that those bounds, computed from the Bézier expansion, are not necessarily sharp. However, they can be sharpened by subdividing, *i.e.* by expanding the same function defined on a smaller domain (called a subdomain). The smaller the subdomain, the sharper the bounds. This subdivision can be implemented in a recursive and adaptive manner which makes the method very efficient [12]. The algorithm for computing sharp bounds on polynomial functions consists of four steps:

1. Sampling of the function on a given set of points.

2. Transformation of those values into Bézier coefficients (by a matrix-vector product).

[1] Let $\boldsymbol{\xi}_c$ be the reference coordinates of one of those corners. For any Bézier basis, there exists an index $\boldsymbol{j}$ such that $B^\nu_{\boldsymbol{j}}(\boldsymbol{\xi}_c) = 1$ and $B^\nu_{\boldsymbol{k}}(\boldsymbol{\xi}_c) = 0,\ \forall \boldsymbol{k} \in \mathcal{I}^\nu \setminus \{\boldsymbol{j}\}$. We have thus the following equality: $f(\boldsymbol{\xi}_c) = f_{\boldsymbol{j}}$.

3. Computation of the bounds. If the sharpness is reached, return the bounds.

4. Subdivision (through a matrix-vector product). For each subdomain, go to step 3. Gather the "subbounds" and compute and return the global bounds.

We propose to adapt this algorithm to the computation of bounds of R. As it will be seen later, only the third step has to be adapted. Additional properties of the Bézier expansion will be needed and are given in the following.

Property 1. *Any Bézier function (even pyramidal) is of the canonical form* $B^\nu_{\boldsymbol{i}}(\boldsymbol{\xi}) = \alpha^\nu_{\boldsymbol{i}}\, m^\nu_{\boldsymbol{i}}(\boldsymbol{\xi})$*, where the coefficient* $\alpha^\nu_{\boldsymbol{i}}$ *is a product of binomial coefficients and* $m^\nu_{\boldsymbol{i}}$ *is the* elementary *function.*

As an example, let us consider the triangular Bézier functions:

$$T^\nu_{i_1,i_2}(\xi,\eta) = \binom{\nu}{i_1}\binom{\nu-i_1}{i_2}\,\xi^{i_1}\,\eta^{i_2}\,(1-\xi-\eta)^{\nu-i_1-i_2}.$$

Its coefficient is $\alpha^\nu_{i_1,i_2} = \binom{\nu}{i_1}\binom{\nu-i_1}{i_2}$ and its *elementary* function is $m^\nu_{i_1,i_2}(\boldsymbol{\xi}) = \xi^{i_1}\,\eta^{i_2}\,(1-\xi-\eta)^{\nu-i_1-i_2}$.

Property 2. *The product of two* elementary *functions* $m^\nu_{\boldsymbol{i}}$ *and* $m^\mu_{\boldsymbol{j}}$ *is an* elementary *function equal to* $m^{\nu+\mu}_{\boldsymbol{i}+\boldsymbol{j}}$.

Example:

$$m^\nu_{i_1,\,i_2}\; m^\mu_{j_1,\,j_2} = \xi^{i_1+j_1}\,\eta^{i_2+j_2}\,(1-\xi-\eta)^{\nu+\mu-(i_1+j_1)-(i_2+j_2)} = m^{\nu+\mu}_{i_1+j_1,\,i_2+j_2}.$$

Corollary 3. *The product of two Bézier functions of order* ν *and* μ *is a Bézier function of order* $\nu+\mu$ *with an adjustment coefficient:*

$$B^\nu_{\boldsymbol{i}}\, B^\mu_{\boldsymbol{j}} = \frac{\alpha^\nu_{\boldsymbol{i}}\alpha^\mu_{\boldsymbol{j}}}{\alpha^{\nu+\mu}_{\boldsymbol{i}+\boldsymbol{j}}}\, B^{\nu+\mu}_{\boldsymbol{i}+\boldsymbol{j}}.$$

Proposition 4. *Let* f *and* g *be two polynomial functions of respective order* ν *and* μ *and let* $f_{\boldsymbol{i}},\ \boldsymbol{i}\in\mathcal{I}^\nu$ *and* $g_{\boldsymbol{j}},\ \boldsymbol{j}\in\mathcal{I}^\mu$ *be their respective control values. The product of* f *and* g *is a polynomial function of order* $\nu+\mu$ *whose control values* $h_{\boldsymbol{k}},\ \boldsymbol{k}\in\mathcal{I}^{\nu+\mu}$ *are equal to:*

$$h_{\boldsymbol{k}} = \sum_{\substack{\boldsymbol{i}\in\mathcal{I}^\nu\\ \boldsymbol{j}\in\mathcal{I}^\mu\\ \boldsymbol{i}+\boldsymbol{j}=\boldsymbol{k}}} f_{\boldsymbol{i}}\, g_{\boldsymbol{j}}\, \frac{\alpha^\nu_{\boldsymbol{i}}\alpha^\mu_{\boldsymbol{j}}}{\alpha^{\nu+\mu}_{\boldsymbol{k}}}.$$

Proof. Using the definition of the Bézier expansion and Corollary 3, we have the following equalities:

$$h = fg = \sum_{\substack{\boldsymbol{i}\in\mathcal{I}^\nu \\ \boldsymbol{j}\in\mathcal{I}^\mu}} f_{\boldsymbol{i}}\, g_{\boldsymbol{j}}\, \frac{\alpha_{\boldsymbol{i}}^\nu \alpha_{\boldsymbol{j}}^\mu}{\alpha_{\boldsymbol{i}+\boldsymbol{j}}^{\nu+\mu}}\, B_{\boldsymbol{i}+\boldsymbol{j}}^{\nu+\mu} = \sum_{\boldsymbol{k}\in\mathcal{I}^{\nu+\mu}} \sum_{\substack{\boldsymbol{i}\in\mathcal{I}^\nu \\ \boldsymbol{j}\in\mathcal{I}^\mu \\ \boldsymbol{i}+\boldsymbol{j}=\boldsymbol{k}}} f_{\boldsymbol{i}}\, g_{\boldsymbol{j}}\, \frac{\alpha_{\boldsymbol{i}}^\nu \alpha_{\boldsymbol{j}}^\mu}{\alpha_{\boldsymbol{k}}^{\nu+\mu}}\, B_{\boldsymbol{k}}^{\nu+\mu}.$$

□

Proposition 5. *Relaxation: Let f and g be two polynomial functions of order ν and let $f_{\boldsymbol{i}}$ and $g_{\boldsymbol{i}}$, $\boldsymbol{i}\in\mathcal{I}^\nu$ be their respective control values. In order that $f(\boldsymbol{\xi}) \le g(\boldsymbol{\xi})$, $\forall\boldsymbol{\xi}\in\Omega_{ref}$, it is sufficient that $f_{\boldsymbol{i}} \le g_{\boldsymbol{i}}$, $\forall\boldsymbol{i}\in\mathcal{I}^\nu$.*

Proof. Since every Bézier function is positive,

$$f_{\boldsymbol{i}} \le g_{\boldsymbol{i}} \quad \Rightarrow \quad f_{\boldsymbol{i}}\, B_{\boldsymbol{i}}^\nu \le g_{\boldsymbol{i}}\, B_{\boldsymbol{i}}^\nu \qquad \forall\boldsymbol{i}\in\mathcal{I}^\nu.$$

□

The last proposition can be used to compute bounds on more complex functions. The following sections explain how to do it for rational functions and functions that are similar to a quadratic mean, *i.e.* that are written $\sqrt{f_1^2 + f_2^2 + \dots}$ where the f_k are polynomial functions.

3.1 Computing bounds on rational functions

It is possible to compute bounds on rational functions whose denominator is strictly positive or strictly negative. There are four cases depending if we want to compute a lower or an upper bound and if the denominator is positive or negative. In the following, we detail the method for computing a lower bound in the case of a strictly positive denominator. The adaptation for other cases is straightforward.

Let $\frac{f}{g}$ be the rational function, with f and $g > 0$ two polynomial functions. Let $f_{\boldsymbol{i}}$ and $g_{\boldsymbol{i}}$, $\boldsymbol{i}\in\mathcal{I}^\nu$ be the control values of their respective Bézier expansion and let r_m denote the lower bound we want to compute. Since g is strictly positive, we can write that the lower bound has to satisfy $r_m\, g \le f$. Taking advantage of the *relaxation* (Proposition 5), we can compute a bound that satisfies the following inequalities:

$$r_m\, g_{\boldsymbol{i}} \le f_{\boldsymbol{i}} \quad \forall\boldsymbol{i}\in\mathcal{I}^\nu.$$

Note that the $g_{\boldsymbol{i}}$'s can take negative values even if g is strictly positive. We have to differentiate several cases. Every inequality for which $g_{\boldsymbol{i}} = 0$

and $f_{\boldsymbol{i}} \geq 0$ is satisfied whatever the value of r_m. On the other hand, the inequalities for which $g_{\boldsymbol{i}} = 0$ and $f_{\boldsymbol{i}} < 0$ cannot be satisfied at all. If this happens, the lower bound cannot be computed using this technique.

When $g_{\boldsymbol{i}}$ is different from zero, it can be passed on the right-hand side and we have $r_m \leq f_{\boldsymbol{i}}/g_{\boldsymbol{i}}$. In function of the sign of the fraction, the inequality constitutes an upper bound or a lower bound for r_m. Let us define $\mathcal{S}^+$ (resp. $\mathcal{S}^-$) as the set of indices $\boldsymbol{i}$ for which $g_{\boldsymbol{i}}$ is strictly positive (resp. strictly negative). We have $r_m^{\text{UB}} = \min_{\boldsymbol{i}\in\mathcal{S}^+} f_{\boldsymbol{i}}/g_{\boldsymbol{i}}$ and $r_m^{\text{LB}} = \max_{\boldsymbol{i}\in\mathcal{S}^-} f_{\boldsymbol{i}}/g_{\boldsymbol{i}}$ [2]. Again, there are two cases to differentiate. When $r_m^{\text{LB}} > r_m^{\text{UB}}$, it is not possible to find a value for r_m that satisfies the upper and lower bound and this technique fails. In the other case, r_m is set to the best possible value, *i.e.* r_m^{UB}.

3.2 Computing bounds on quadratic mean-like functions

Computing bounds with the square root relies on the following lemma:

Lemma 6. *Let f and g be two polynomial functions of order ν and let $f_{\boldsymbol{i}}$ and $g_{\boldsymbol{i}}$ be their control values. Let H be the function whose control values are $H_{\boldsymbol{i}} = \sqrt{f_{\boldsymbol{i}}^2 + g_{\boldsymbol{i}}^2}$. Then, we have $\sqrt{f(\boldsymbol{\xi})^2 + g(\boldsymbol{\xi})^2} \leq H(\boldsymbol{\xi})\ \forall \boldsymbol{\xi}$.*

Proof. Since the two members of the inequality are positive, we can square the expression:

$$\left(\sum_{\boldsymbol{i}} f_{\boldsymbol{i}} B_{\boldsymbol{i}}^{\nu}(\boldsymbol{\xi})\right)^2 + \left(\sum_{\boldsymbol{i}} g_{\boldsymbol{i}} B_{\boldsymbol{i}}^{\nu}(\boldsymbol{\xi})\right)^2 \leq \left(\sum_{\boldsymbol{i}} \sqrt{f_{\boldsymbol{i}}^2 + g_{\boldsymbol{i}}^2}\ B_{\boldsymbol{i}}^{\nu}(\boldsymbol{\xi})\right)^2$$

$$\Leftrightarrow \sum_{\boldsymbol{i},\boldsymbol{j}} (f_{\boldsymbol{i}}\, f_{\boldsymbol{j}} + g_{\boldsymbol{i}}\, g_{\boldsymbol{j}})\, B_{\boldsymbol{i}}^{\nu}(\boldsymbol{\xi}) B_{\boldsymbol{j}}^{\nu}(\boldsymbol{\xi}) \leq \sum_{\boldsymbol{i},\boldsymbol{j}} \sqrt{f_{\boldsymbol{i}}^2\, f_{\boldsymbol{j}}^2 + f_{\boldsymbol{i}}^2\, g_{\boldsymbol{j}}^2 + g_{\boldsymbol{i}}^2\, f_{\boldsymbol{j}}^2 + g_{\boldsymbol{i}}^2\, g_{\boldsymbol{j}}^2}\, B_{\boldsymbol{i}}^{\nu}(\boldsymbol{\xi}) B_{\boldsymbol{j}}^{\nu}(\boldsymbol{\xi}).$$

By the relaxation of Proposition 5, it is sufficient to prove the following inequalities:

$$f_{\boldsymbol{i}}\, f_{\boldsymbol{j}} + g_{\boldsymbol{i}}\, g_{\boldsymbol{j}} \leq \sqrt{f_{\boldsymbol{i}}^2\, f_{\boldsymbol{j}}^2 + f_{\boldsymbol{i}}^2\, g_{\boldsymbol{j}}^2 + g_{\boldsymbol{i}}^2\, f_{\boldsymbol{j}}^2 + g_{\boldsymbol{i}}^2\, g_{\boldsymbol{j}}^2} \quad \forall \boldsymbol{i}, \boldsymbol{j}.$$

The right-hand member is positive. This implies that the relation is satisfied if the relation hold when the members are squared. We obtain:

$$\begin{aligned} 2\, f_{\boldsymbol{i}}\, f_{\boldsymbol{j}}\, g_{\boldsymbol{i}}\, g_{\boldsymbol{j}} &\leq f_{\boldsymbol{i}}^2\, g_{\boldsymbol{j}}^2 + g_{\boldsymbol{i}}^2\, f_{\boldsymbol{j}}^2 \quad \forall \boldsymbol{i}, \boldsymbol{j} \\ \Leftrightarrow 0 &\leq (f_{\boldsymbol{i}}\, g_{\boldsymbol{j}} - g_{\boldsymbol{i}}\, f_{\boldsymbol{j}})^2 \quad \forall \boldsymbol{i}, \boldsymbol{j}, \end{aligned}$$

[2] Note that the sets $\mathcal{S}^-$ and/or $\mathcal{S}^+$ can be empty in which cases we have $\min_{\boldsymbol{i}\in\emptyset} f_{\boldsymbol{i}}/g_{\boldsymbol{i}} = +\infty$ and $\max_{\boldsymbol{i}\in\emptyset} f_{\boldsymbol{i}}/g_{\boldsymbol{i}} = -\infty$.

which is true. □

This allows to compute an upper bound on the square root that is equal to the maximum control value of H. Note that this lemma can be generalized to any function of the form $\sqrt{f_1^2 + f_2^2 + \dots}$ where the f_k are polynomial functions.

4 Computing bounds of the measure in 2D

Here we first obtain the expression of the eigenvalues of the metric in function of quantities that can be sampled on the element. We identify the polynomial functions for which the steps 1, 2 and 4 of the algorithm can be directly applied (see Section 3). Then we explain how to compute the bounds of $R_{\min}$ in replacement of the 3rd step.

Let us consider that $\boldsymbol{x}(\boldsymbol{\xi})$ defines the mapping between the ideal and the physical element. The 2D metric tensor reads

$$\mathcal{M}(\boldsymbol{\xi}) := \begin{pmatrix} ||\boldsymbol{x}_{,\xi}||^2 & \boldsymbol{x}_{,\xi} \cdot \boldsymbol{x}_{,\eta} \\ \boldsymbol{x}_{,\xi} \cdot \boldsymbol{x}_{,\eta} & ||\boldsymbol{x}_{,\eta}||^2 \end{pmatrix}.$$

The expression of the eigenvalues are easily computed from the characteristic equation. By resolving the second order equation and after simplification, we obtain:

$$\lambda_{\pm}(\boldsymbol{\xi}) = \frac{||\boldsymbol{x}_{,\xi}||^2 + ||\boldsymbol{x}_{,\eta}||^2}{2} \pm \sqrt{\left(\frac{||\boldsymbol{x}_{,\xi}||^2 - ||\boldsymbol{x}_{,\eta}||^2}{2}\right)^2 + (\boldsymbol{x}_{,\xi} \cdot \boldsymbol{x}_{,\eta})^2}.$$

Due to the presence of the square root, they are not polynomials. However, $\boldsymbol{x}_{,\xi}$ and $\boldsymbol{x}_{,\eta}$ are polynomials and so are their norm and their scalar product. The above expression can be written to let appear three polynomial functions q, s and t:

$$\lambda_{\pm}(\boldsymbol{\xi}) = q(\boldsymbol{\xi}) \pm \sqrt{s(\boldsymbol{\xi})^2 + t(\boldsymbol{\xi})^2}.$$

The first function, q, is greater or equal to 0 while the two others take their values in $\mathbb{R}$. One can calculate the extrema of the square root and find that $\sqrt{s^2 + t^2} \in [0, q]$ which is in accordance with the fact that the eigenvalues of $\mathcal{M}$ are positive or zero. The functions q, s and t are of order $2n$, where n is the order of the element. They can be expanded into a Bézier basis of order $2n$. The steps 1, 2, and 4 of the algorithm given in Section 3 can be applied to these functions. It remains to adapt the third step, *i.e.* to find a way to compute a bound of the quality from the control values.

We note $q_{\boldsymbol{i}}$, $s_{\boldsymbol{i}}$ and $t_{\boldsymbol{i}}$ the control values of the polynomial functions. Since it is possible to compute an upper bound for the square root (see Sect. 3.2), the two following bounds can be computed:

$$\lambda_-^{\mathrm{LB}} = \min_{\boldsymbol{i}} \left[q_{\boldsymbol{i}} - \sqrt{s_{\boldsymbol{i}}^2 + t_{\boldsymbol{i}}^2} \right] \quad \text{and} \quad \lambda_+^{\mathrm{UB}} = \max_{\boldsymbol{i}} \left[q_{\boldsymbol{i}} + \sqrt{s_{\boldsymbol{i}}^2 + t_{\boldsymbol{i}}^2} \right].$$

Note that the eigenvalues are positive but λ_-^{LB} can sometimes be negative in which case it is set to zero. Those bounds are useful if one want to get bounds on the eigenvalues alone. For the quality measure we consider (defined in Equation (**??**)), we have to compute a lower and an upper bound of $R_{\min} := \min_{\boldsymbol{\xi} \in \Omega_{\mathrm{ref}}} R(\boldsymbol{\xi})$. The upper bound is computed from the corner values:

$$R_{\min}^{\mathrm{UB}} = \min_{\boldsymbol{i} \in \mathcal{I}_c^\nu} \frac{q_{\boldsymbol{i}} - \sqrt{s_{\boldsymbol{i}}^2 + t_{\boldsymbol{i}}^2}}{q_{\boldsymbol{i}} + \sqrt{s_{\boldsymbol{i}}^2 + t_{\boldsymbol{i}}^2}}.$$

A lower bound is given by the ratio $\lambda_-^{\mathrm{LB}}/\lambda_+^{\mathrm{UB}}$ but it is easy to understand that this bound is poor since the point at which λ_- takes its minimum is in general not the same that the point at which λ_+ takes its maximum. In order to obtain a sharp bound, we introduce $a(\boldsymbol{\xi}) = \frac{q}{\sqrt{s^2+t^2}} \in [1, +\infty[$ so that we can write :

$$R(a) = \frac{a-1}{a+1}.$$

Thankfully, this function increases monotonically which implies that $R_{\min} = R(a_{\min})$. We see that we can compute $R_{\min}^{\mathrm{LB}} = R(a_{\min}^{\mathrm{LB}})$ where $a_{\min}^{\mathrm{LB}}$ is the bound computed. The method to bound rational functions (see Section 3.1) cannot be applied to a since the denominator is a square root. However, since a is positive, we have the following equality $\min_{\boldsymbol{\xi}} a^2 = [\min_{\boldsymbol{\xi}} a]^2$. We can thus compute a lower bound of a^2 and take its square root.

The numerator, q^2, and the denominator, $s^2 + t^2$, are expanded into the Bézier basis of order $4n$ by relying on Proposition 4. This allows to apply the method to bound rational functions. Let r_m be the computed bound. If the method fails to provide a bound, r_m is simply set to 1. At the end, the returned lower bound of a is $\sqrt{\max(1, r_m)}$.

5 Computing bounds of the measure in 3D

Like in 2D, we first derive the expression of the eigenvalues, then we explain how to compute the bounds of $R_{\min}$.

The characteristic equation of a 3×3 matrix is a cubic equation that is not easily solved. Thankfully, the metric tensor is symmetric and a simple expression can be obtained for the eigenvalues. The key is to make an affine transform that *in fine* leads to a trigonometric solution [22]. Let us introduce three functions, q, p and ϕ: $q(\boldsymbol{\xi}) := \mathrm{tr}(\mathcal{M})/3$, $p(\boldsymbol{\xi}) := \sqrt{\mathrm{tr}[(\mathcal{M} - qI)^2]\,/\,6}$ and $\phi(\boldsymbol{\xi}) \in [0, \frac{\pi}{3}]$. Their complete expression is given in Appendix A. The eigenvalues of $\mathcal{M}$ are written:

$$\lambda_k(\boldsymbol{\xi}) = q(\boldsymbol{\xi}) + 2\,p(\boldsymbol{\xi}) \cos\left(\phi(\boldsymbol{\xi}) + k\frac{2\pi}{3}\right), \qquad k \in \{0, 1, 2\},$$

with $(\lambda_- =)\, \lambda_1 \leq \lambda_2 \leq \lambda_0\, (= \lambda_+)$. Since the metric tensor is real and symmetric, its eigenvalues are also real. The function q is a positive polynomial function and is the equivalent of the 2D function q. The function p is the equivalent of the function $\sqrt{s^2 + t^2}$. Each term of the square root is polynomial and, as in 2D, $p \in [0, q]$. A more convenient expression can be obtained for ϕ. Let J denote the Jacobian determinant. Its square is equal to the product of the eigenvalues: $\lambda_0\lambda_1\lambda_2 = \det(\mathcal{M}) = \det(W^T W) = J^2$, and by replacing the eigenvalues by their expression, one can get the following formula:

$$\phi = \frac{1}{3}\arccos\left(\frac{1}{2}\left[\frac{J^2 - q^3 + 3p^2 q}{p^3}\right]\right).$$

The functions J, p and q are thus the polynomial functions that are used for steps 1,2 and 4 of the algorithm. Due to the presence of the cosine and the inverse cosine, computing bounds on the 3D eigenvalues alone is much more complex than in 2D. However, we can still compute bounds on R due to a substantial simplification it brings. Similarly to what we did in 2D, let us introduce $a(\boldsymbol{\xi}) = q/p \in [1, +\infty[$. Let us also introduce $K(\boldsymbol{\xi}) = J^2/p^3 \in \mathbb{R}^+$. Now, the ratio λ_-/λ_+ reads:

$$R(a, K) = \frac{a + 2\cos(\phi + \frac{2\pi}{3})}{a + 2\cos(\phi)},$$

where $\phi = 1/3\arccos(w)$, with $w = 1/2\left[K - a^3 + 3a\right]$. Figure 2 shows the graph of $R(a, K)$. Its domain of definition is defined by the existence of ϕ. Mathematically, we have $\Omega_R = \{(a, K) \in [1, +\infty[\times\mathbb{R}^+ : w(a, K) \in [-1, 1]\}$. Note that the denominator is strictly positive since $\cos(\phi) \in [\frac{1}{2}, 1]$, thus R is well defined.

Let us clarify those definitions: we have two mappings. The first associates a point $\big(a(\boldsymbol{\xi}), K(\boldsymbol{\xi})\big) \in \Omega_R$ to each point $\boldsymbol{\xi} \in \Omega_{\mathrm{ref}}$ of the reference element. This mapping depends on the node position of the physical element.

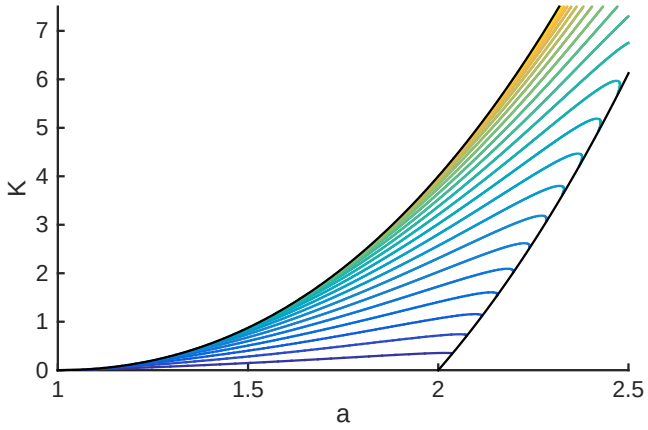


Figure 2: $R(a, K)$: Its domain of definition is limited by the black curves and its value is represented by isovalues.

The second is element-independent and associates a value $R(a, K) \in [0, 1]$ to each point $(a, K) \in \Omega_R$. We see that $R_{\min}$ can theoretically be computed in two steps: (1) computation of Ω_{aK}, the region of the aK-plane that the element *affects*, *i.e.* $\Omega_{aK} = \left\{\left(a(\boldsymbol{\xi}), K(\boldsymbol{\xi})\right) \in \Omega_R : \boldsymbol{\xi} \in \Omega_{\text{ref}}\right\}$ and (2) computation of the minimum of R restricted to the domain Ω_{aK}. It is proved in Appendix C that this minimum is located on the boundary of Ω_{aK}. Yet, computing this boundary is a complex problem that is certainly too costly to be interesting in practice. The strategy we use instead is to compute a *bounding box* $\tilde{\Omega}$ that includes Ω_{aK} and for which it is easy to compute the minimum of R.

The bounding box is constructed thanks to the lower and upper bounds of a and K. Figure 3 shows an example of Ω_{aK} and a possible bounding box. Let a_m and K_m denote the lower bounds and a_M and K_M the upper bounds. Computing a_m is done exactly the same way than in 2D (Section 4), provided that $\sqrt{s^2 + t^2}$ is replaced by p. The adaptation of this technique for computing a_M does not present any difficulty, so the details are omitted for brevity. In the following subsections, we first explain how to compute the bounds of K. Then we explain how to determine the minimum of R inside the bounding box and finally, we explain the limitations of the bounding box approach.

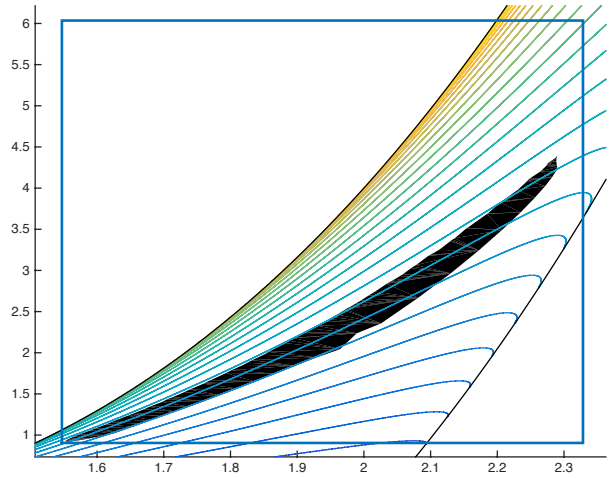


Figure 3: The region that the element affects, Ω_{aK} (in black) can be bounded by a box (in blue).

5.1 Computing a lower bound of K

The technique to compute a lower bound for K is very similar to the technique of Section 4. Let $p_{\boldsymbol{i}}^{(k)},\ k \in \{1,\dots,6\},\ \boldsymbol{i} \in \mathcal{I}^{2n}$ be the control values of the six terms of p. We define the following function:

$$P(\boldsymbol{\xi}) := \sum_{\boldsymbol{i} \in \mathcal{I}^{2n}} P_{\boldsymbol{i}}\ B_{\boldsymbol{i}}^{2n}(\boldsymbol{\xi}), \quad \text{with} \quad P_{\boldsymbol{i}} := \sqrt{\sum_{k=1}^{6} \left[p_{\boldsymbol{i}}^{(k)}\right]^2}.$$

Due to the generalization of Lemma 6, we have $p(\boldsymbol{\xi}) \leq P(\boldsymbol{\xi})\ \forall \boldsymbol{\xi} \in \Omega_{\text{ref}}$. This implies that $K \geq J^2/P^3$ and if we compute a lower bound for J^2/P^3, it will be a lower bound for K. In order to do that, we apply the method for computing bounds on rational functions. The numerator is expanded into the Bézier basis of order $6n$ by applying the Proposition 4. The denominator is a multiplication of three functions but Proposition 4 can easily be adapted to this case. We define $\sigma_{\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}}^{\nu} := \alpha_{\boldsymbol{i}}^{\nu}\alpha_{\boldsymbol{j}}^{\nu}\alpha_{\boldsymbol{k}}^{\nu}/\alpha_{\boldsymbol{i}+\boldsymbol{j}+\boldsymbol{k}}^{3\nu}$ and the set $\mathcal{E}_3^{\nu}(\boldsymbol{l}) := \{(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}),\ \boldsymbol{i},\boldsymbol{j},\boldsymbol{k} \in \mathcal{I}^{\nu} : \boldsymbol{i}+\boldsymbol{j}+\boldsymbol{k} = \boldsymbol{l}\}$. The denominator is thus expanded into the Bézier basis of order $6n$ with the following control values:

$$P'_{\boldsymbol{l}} = \sum_{(\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}) \in \mathcal{E}_3^{2n}(\boldsymbol{l})} P_{\boldsymbol{i}}P_{\boldsymbol{j}}P_{\boldsymbol{k}}\ \sigma_{\boldsymbol{i},\boldsymbol{j},\boldsymbol{k}}^{2n}.$$

Finally, the lower bound is computed by using the technique for rational functions (Section 3.1). If the technique fails, the lower bound is set to 0.

5.2 Computing an upper bound of K

If we compute an upper bound for J^2/P^3, it would not necessarily be an upper bound for K since we have $K \geq J^2/P^3$. There are two other possibilities. We can either compute $(J^2)^{\mathrm{UB}}/(p^3)^{\mathrm{LB}}$ or compute an upper bound for K^2 by applying the same technique as in Section 4. The latter would be sharper but it would require to expand the functions into the Bézier basis of order $12p$. The number of coefficients to compute would be proportional to this order cubed as we have $|\mathcal{I}^\nu| = O(\nu^3)$ for 3D elements (where $|\cdot|$ stands for the cardinality of a set). In consequence, it would be more costly to compute this bound than the others, whereas it is not the most important. We thus prefer to compute the first one, *i.e.* $(J^2)^{\mathrm{UB}}/(p^3)^{\mathrm{LB}}$.

For the numerator, it can be easily demonstrated that $\max_{\boldsymbol{i}\in\mathcal{I}^{3p}} J^2_{\boldsymbol{i}}$ constitutes an upper bound of J^2. Computing a lower bound on p^3 relies on the fact that if a and b are positive, then $a \leq b \Leftrightarrow a^\nu \leq b^\nu$ if ν is a real positive number. We can then compute a bound r_m that satisfies $0 \leq r_m \leq p^2$ for which we know that $r_m^{3/2}$ will be a lower bound of p^3. By expanding p^2 into the Bézier basis of order $4n$ and by *relaxing*, we obtain:

$$r_m^\star = \min_{\boldsymbol{k}\in\mathcal{I}^{4n}} \sum_{(\boldsymbol{i},\boldsymbol{j})\,\in\,\mathcal{E}_2^{2n}(\boldsymbol{k})} \left[\sum_{l=1}^{6} p_{\boldsymbol{i}}^{(l)}\, p_{\boldsymbol{j}}^{(l)}\right] \frac{\alpha_{\boldsymbol{i}}^\nu \alpha_{\boldsymbol{j}}^\nu}{\alpha_{\boldsymbol{i}+\boldsymbol{j}}^{2\nu}}$$

and $r_m = \max(0,\, r_m^\star)$.

5.3 Computing the minimum of R inside the bounding box

Finding the location of the minimum relies on the gradient of R and the first derivatives of R are given in Appendix B. As can be deduced from Figures 2 and 4, it can be proved that dR/da is zero on a curved line, is negative on the left of this line and positive on the right. Similarly, it can be proved that dR/dK is zero on another curved line (located to the right of the previous one), is negative below and positive above. This implies that there are 5 possible locations for the minimum of R (see Figure 4). Computing the first derivative of R at the corners of the bounding box allows to determine the location. Alternatively, one can compute the zero of $\frac{dR}{da}(a, K_m)$, which we refer as a_0, and the zero of $\frac{dR}{dK}(a_m, K)$, which we refer as K_0, and compare those quantities to the bounds of the box. As an example let us consider

the fourth case: it arises when $K_m < K_0 < K_M$ (and $a_0 < a_m \leq a_M$). In this case, the solution is $R^{\text{LB}}_{\text{min}} = R(a_m, K_0)$.

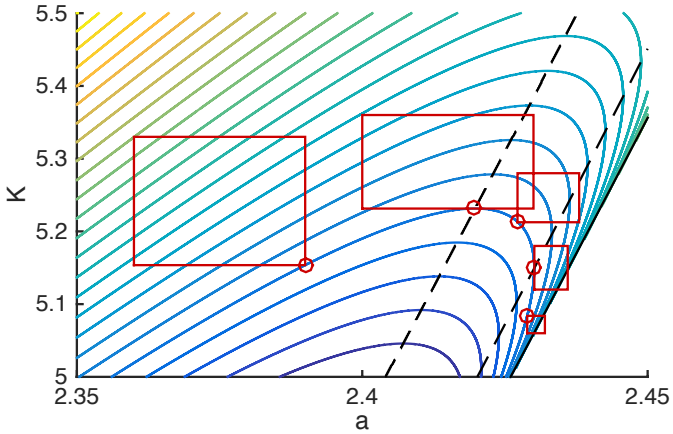


Figure 4: There exists 5 different cases for the location of the minimum of R inside the bounding box.

The expression of K_0 can be calculated and is the following:

$$K_0 = 2\cos\left(3\arccos\left(-\frac{1}{a_m}\right) - \pi\right) + a_m^3 - 3\,a_m.$$

However, due to the complex expression of $\frac{dR}{da}$, a_0 is computed numerically.

5.4 Limitations of the bounding box strategy

Let us consider the example of the tetrahedron and its corresponding $\Omega_{a,K}$ shown in Figure 5. We can see that $\Omega_{a,K}$ is very elongated, nearly in the direction of the isovalues of R. Although the element is only slightly curved ($R(\boldsymbol{\xi})$ is between 0.2186 and 0.2462), the lower bound computed from the bounding box technique is 0.1044 which is not sharp at all. Obviously, a sharp bound will be obtained by subdivision (step 4 of the algorithm), however, it will be at the cost of high computational efforts. In particular, $20,899$ subdivisions and a running time of $1,32$ seconds is necessary to compute the measure at a tolerance of 10^{-2}. This is because each subdivision consists in splitting the tetrahedron into 8 sub-tetrahedra for which the bounds have to be computed and every subdomain must be subdivided again if the tolerance is not reached. This case is not rare and shows that the subdivision technique can lead to an enormous number of subdomains if poor bounds are provided. As will be shown in the results, it is worth

computing better bounds, even if each computation is more expensive, which is the subject of the following section.

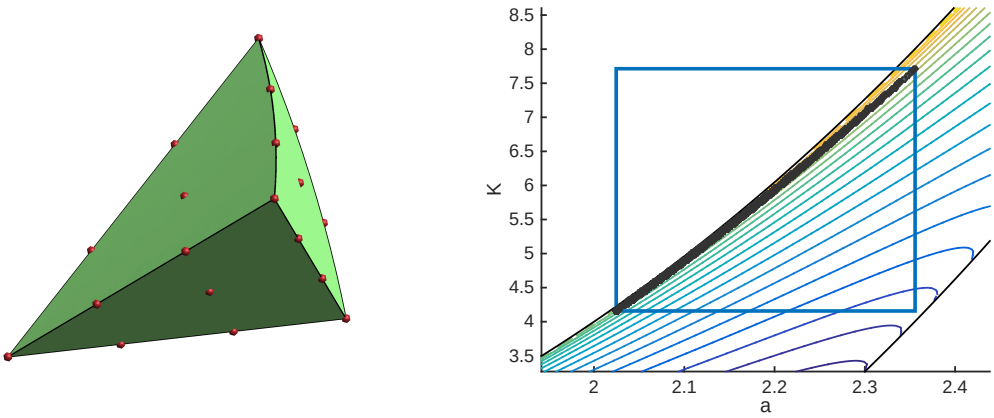


Figure 5: A slightly curved tetrahedra (of order 3) and the corresponding region $\Omega_{a,K}$ that the element affects. The bounding box technique for this kind of element is highly inefficient.

6 A bounding curve for the domain Ω_{aK}

In order to improve the bounds, we seek to compute an additional bounding curve that will cut the bounding box. Many of them can be identified:

1. $K = \kappa\, a$: It is the simplest curve we can think of. It passes through the origin and has a slope κ. The latter is a parameter but it can be seen as the function $\kappa(a, K) = {}^{K}/_{a} = {}^{J^2}/_{qp^2}$ for which the lower bound, κ_m, and the upper bound, κ_M, can easily be computed using the rational function bounding technique. As a result, the two curves $K = \kappa_m a$ and $K = \kappa_M a$ constitute bounding curves for the domain Ω_{aK}. They are however of poor quality because the slope cannot be chosen. They can greatly improve the bound of R by luck but most of the time they will not.

2. $K = \kappa\, a + c$: In order to overcome the drawback of the first proposed curve, we can specify the slope κ and bound $c(a, K) = {}^{(J^2 - \kappa\, qp^2)}/_{p^3}$. Like for computing the lower bound of K (Sect. 5.1), the denominateur can be replaced by P^3 which allows to compute a lower bound if the numerator is strictly positive. If the numerator is strictly negative, then it is an upper bound that can be computed. Moreover, no bound

can be computed if the numerator takes positive and negative values. It is still always possible to find a slope that allows to compute a lower or an upper bound on c (in function of what is needed) but the choice is greatly limited. Like for the upper bound of K (Sect. 5.2), we could consider computing the bound differently but either the bound would be poor or the computation would be time consuming. The best choice is to consider the next curve.

3. $K = \beta\, a^3 + \kappa\, a$: This curve have the advantages of the two preceding curves without the disadvantages. Indeed, since there are two parameters, we have a control on the slope of the curve. Morevover, by specifying κ, we have to compute bounds on $\beta(a, K) = {}^{(J^2 - \kappa\, qp^2)}/_{q^3}$ which is as easily done as for the first curve.

It is proved in Appendix D that the function $R(a, K)|_{K=\beta a^3+\kappa a}$ admits a unique global minimum. In addition to the bounding box, the new algorithm computes the parameter $\kappa^\star$ such that the curve that passes through the bottom left corner of the bounding box have the same slope that the isovalues of R; this way the minimum of $R(a, K)|_{K=\beta a^3+\kappa^\star a}$ would be located at the bottom left corner in this case. Then a lower bound $\beta_{\min}$ of ${}^{(J^2 - \kappa^\star\, qp^2)}/_{q^3}$ is computed. The curve $K = \beta_{\min}\, a^3 + \kappa^\star\, a$ constitutes the bounding curve. In order to locate the minimum of R in the new *bounding region*, the slope of $R(a, K)|_{K=\beta a^3+\kappa^\star a}$ at the intersections of the bounding curve with the bounding box is computed. If the function increases when going inside the bounding box, then the minimum of R is located on the bounding box and is computed as previously explained. If the function decreases when going inside the bounding box, the minimum of R is computed on the curve using a Newton-Raphson technique.

7 Results

The algorithm described in this paper is implemented in Gmsh and can be tested through the plugin `AnalyseCurvedMesh`. We begin this section by testing two linear hexahedra. Then we test an academic hexahedral mesh with high differences in aspect ratio. After, we test a more realistic mesh composed of curved tetrahedra. All the tests has been conducted on a Macbook Pro Retina, Mid 2012 @ 2.3GHz.

7.1 Single hexahedron test

Let us consider the case of a single hexahedron for which the nodes location is given in Table 2 of reference [9]. This hexahedron has positive Jacobian determinant on all the edges but is invalid. Our algorithm finds the correct value of $\mu = 0$ after one subdivision. Now let us consider a more interesting case of a twisted hexahedron whose nodes location is given in Figure 6. This

i	x_i	y_i	z_i
1	-0.5	-1	-1
2	1	-1	-1
3	1	1	-1
4	-0.5	1	-1
5	1	-1	1
6	1	1	1
7	-1	1	1
8	-1	-1	1

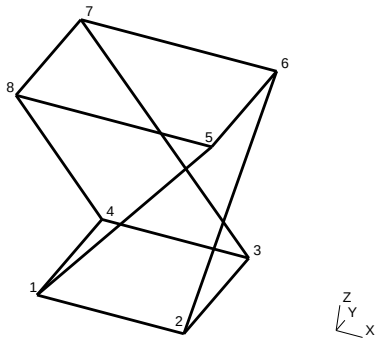


Figure 6: **Twisted hex**: Location of the nodes. Note that the node order is the same than Gmsh.

hexahedron is valid and the measure μ computed at a tolerance of $2e^{-7}$ is in the range $[0.2700452884, 0.2700453384]$. The minimum of μ_{pw} at the corners of the element is 0.3233, which is an error of 0.0533. In order to compare with the basic method which consists in sampling the pointwise measure at a large number of points, let us consider the nodes of an hexahedron of order p. We sample the measure μ_{pw} at the location of those nodes and compute the absolute error. As shown in Figure 7, the error decreases slowly. With $p = 25$, which corresponds to $17,576$ sampling points, the error is still $7e^{-5}$.

7.2 Academic hexahedral mesh

We now test the measure on an academic mesh composed of linear hexahedra. The mesh is generated with a mapping technique and presents high differences in sizes and aspect ratio, see Figure 8. The tested mesh contains $1,000,000$ hexahedra and $1,030,301$ nodes. The measure μ takes 71 seconds to be computed and ranges from 0 to 0.914, while the average is 0.208. As said when we have introduced the measure, it is the distance from the degeneracy of the element and the more an element is anisotropic, the nearer it is to degeneracy. As can be inferred and as can be seen from the worst

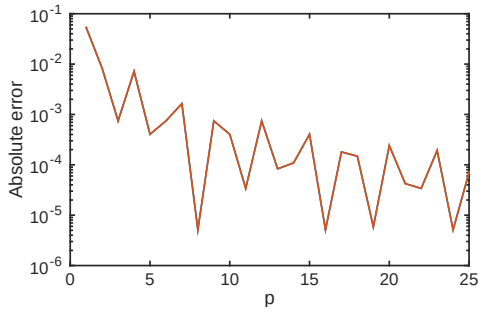


Figure 7: **Twisted hex**: Absolute error of the sampling of μ_{pw} at the nodes of an hexahedron of order p.

elements presented in Figure 9 (left), this measure does not take into account the fact that hexahedra can be elongated in one of its three principal directions without losing accuracy of the finite element solution.

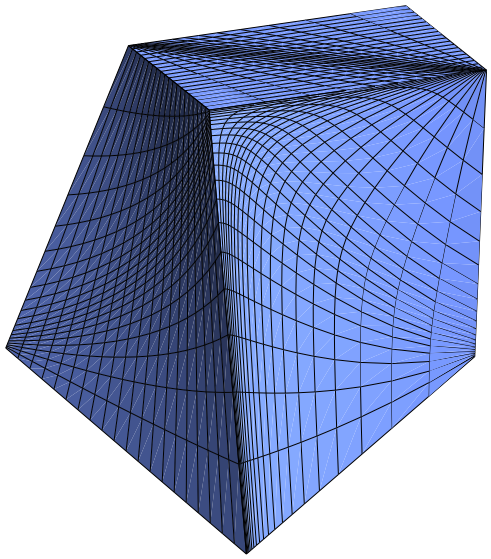


Figure 8: **Hex mesh**: Coarse version (with only 8,000 hexahedra) of the academic 1,000,000 hexahedra test case.

7.3 Realistic curved tetrahedral mesh

Finally, we experiment the measure on a more realistic geometry that is meshed with third-order curved tetrahedra. A coarse and a fine mesh are

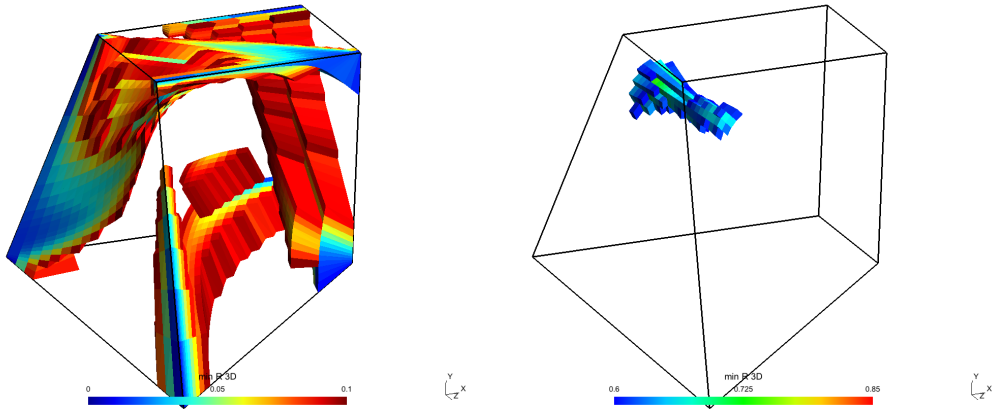


Figure 9: **Hex mesh**: Worst (left) and best (right) quality elements of the coarse mesh.

generated, optimized using the method presented in [8], see Figure 10. The results are given in Table 7.3 and the worst elements are showed in Figure 11.

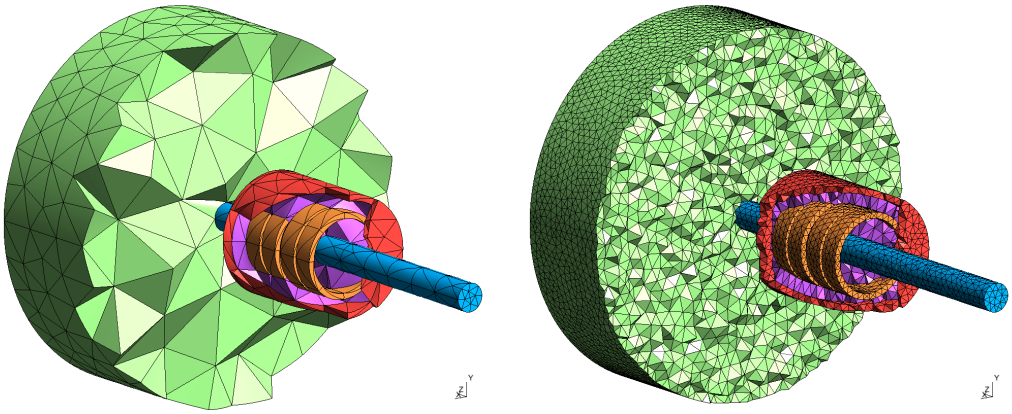


Figure 10: **Curved tetrahedral mesh**: The geometry models a sensor for a steel cable and is composed of 4 coils (in orange), the cable (in blue) and the mounting box (in red). In purple and in green are respectively the interior and the exterior meshes of the voids. The geometry has been meshed with tetrahedra of order 3. The coarse mesh contains 8,252 tetrahedra while the fine mesh contains 293,776 tetrahedra.

Geometry	#elem	#vert	T gen	T optim	T qual	$\mu_{\min}$	μ_{avg}
Magnet., p3	8,252	41,253	2.99	42.03	14.37	0.012	0.348
Magnet., p3	293,776	1,388,750	60.39	-	70.64	0.034	0.55

Table 1: **Curved tetrahedral mesh**: The time are given in seconds. The generation time (T gen) comprises the creation of 1D, 2D and 3D elements, the topological optimization (using Netgen [23]) and the curving of entities that are classified on the boundaries. Note that no optimization was needed for the fine mesh.

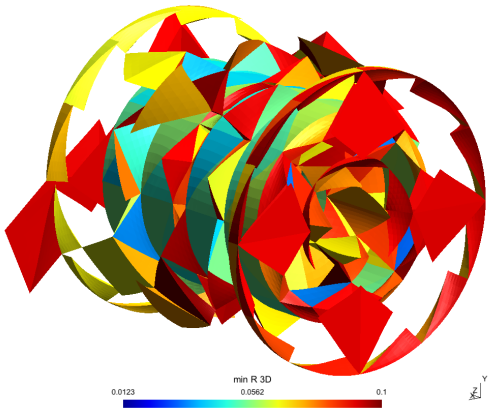


Figure 11: **Curved tetrahedral mesh**: Worst elements of the coarse mesh which are unsurprisingly located in the mounting box.

8 Conclusion

A method for computing the minimum of a pointwise shape quality measure defined for any order and any type of common finite elements (including pyramids) has been presented. The measure is based on the Jacobian matrix and has a mathematical interpretation: it gives the distance of the element to the degeneracy. Geometrically, it is a measure of the anisotropy of the element. The computation is efficient and numerical experiments show that for realistic third-order tetrahedral meshes the computation time of the quality measure is of the same order as the mesh generation time.

The measure is useful for hexahedra as long as isotropic meshes are considered, or if the anisotropy can be considered constant over the element. It is not however a good quality measure for curved high aspect ratio hexahedra, as for example in boundary layer meshes high curvature. A possible

extension of the present method in such cases would then be to compute the minimum of the scaled Jacobian.

A Expression of q, p and ϕ for the 3D eigenvalues

$$q(\boldsymbol{\xi}) = \frac{||\boldsymbol{x}_{,\xi}||^2 + ||\boldsymbol{x}_{,\eta}||^2 + ||\boldsymbol{x}_{,\zeta}||^2}{3}$$

$$p(\boldsymbol{\xi}) = \frac{1}{\sqrt{6}} \sqrt{\begin{array}{l} \left(||\boldsymbol{x}_{,\xi}||^2 - q\right)^2 + \left(||\boldsymbol{x}_{,\eta}||^2 - q\right)^2 + \left(||\boldsymbol{x}_{,\zeta}||^2 - q\right)^2 \\ +2\left[(\boldsymbol{x}_{,\xi} \cdot \boldsymbol{x}_{,\eta})^2 + (\boldsymbol{x}_{,\xi} \cdot \boldsymbol{x}_{,\zeta})^2 + (\boldsymbol{x}_{,\eta} \cdot \boldsymbol{x}_{,\zeta})^2\right] \end{array}}$$

$$\phi(\boldsymbol{\xi}) = \frac{1}{3} \arccos\left(\frac{1}{2} \det\left(\frac{\mathcal{M} - qI}{p}\right)\right)$$

We also express p in short as:

$$p = \sqrt{\sum_{k=1}^{6} \left[p^{(k)}\right]^2},$$

with $p^{(1)} = \left[||\boldsymbol{x}_{,\xi}||^2 - q\right] / \sqrt{6}$, ... and $p^{(4)} = \left[\boldsymbol{x}_{,\xi} \cdot \boldsymbol{x}_{,\eta}\right] / \sqrt{3}$, ...

B First and second derivatives of R

Let $A = \frac{1-a^2}{\sqrt{1-w^2}}$ and $C = 1 + a\,\cos\!\left(\phi + \frac{\pi}{3}\right)$. The expression of the first and second derivatives of R are:

$$\frac{\partial R}{\partial K} = \frac{\sqrt{3}}{3\,(a+2\cos\phi)^2}\left[\frac{C}{\sqrt{1-w^2}}\right]$$
$$\frac{\partial R}{\partial a} = \frac{\sqrt{3}}{(a+2\cos\phi)^2}\left[2\sin\Big(\phi+\frac{\pi}{3}\Big)+AC\right]$$
$$\frac{\partial^2 R}{\partial K^2} = \frac{\sqrt{3}}{6\,(a+2\cos\phi)^2\,(1-w^2)}\left[\frac{1}{3}\,a\sin\Big(\phi+\frac{\pi}{3}\Big)-\frac{4}{3}\,\frac{C\sin\phi}{a+2\cos\phi}+\frac{w\,C}{\sqrt{1-w^2}}\right]$$
$$\frac{\partial^2 R}{\partial aK} = \frac{\sqrt{3}}{3(a+2\cos\phi)^2\sqrt{1-w^2}}\left[\frac{aA}{2}\sin\Big(\phi+\frac{\pi}{3}\Big)+\frac{3}{2}\frac{wAC}{\sqrt{1-w^2}}\right.$$
$$\left.+\cos\Big(\phi+\frac{\pi}{3}\Big)-2C\,\frac{1+A\sin\phi}{a+2\cos\phi}\right]$$
$$\frac{\partial^2 R}{\partial a^2} = \frac{\sqrt{3}}{(a+2\cos\phi)^2}\left[\frac{-2aC}{\sqrt{1-w^2}}+\frac{3}{2}\frac{wA^2C}{\sqrt{1-w^2}}+\frac{aA^2}{2}\sin\Big(\phi+\frac{\pi}{3}\Big)\right.$$
$$\left.-\frac{2\,(1+A\sin\phi)}{a+2\cos\phi}\left[2\sin\Big(\phi+\frac{\pi}{3}\Big)+AC\right]\right]$$

C Proof that R takes its extreme values on the boundary of a given region

For any $\phi^\star \in [0, \frac{\pi}{3}]$, let $\Omega_{\phi^\star}$ denote the domain of R restricted to $\phi(a,K) = \phi^\star$. We construct the 1-dimensional function $R_{\phi^\star}(a) = R(a,K)|_{(a,K)\in\Omega_{\phi^\star}}$. It derivative is

$$\frac{dR_{\phi^\star}}{da} = \frac{2\sqrt{3}\sin(\phi^\star+\frac{\pi}{3})}{(a+2\cos(\phi^\star))^2},$$

which involves that the function $R_{\phi^\star}$ increase monotonically with a. As a consequence, the extreme value that R takes on $\Omega_{aK}\cap\Omega_{\phi^\star}$ are exclusively located on the intersection of the boundary of Ω_{aK} with $\Omega_{\phi^\star}$. It is true for every $\phi^\star$ which prove that the extrema of R inside Ω_{aK} are to be found on the boundary.

D Proof that the function R restricted to the bounding curve admits a unique minimum

Let us consider the curve $\mathcal{C}^{\beta,\kappa} \equiv K = \beta a^3 + \kappa a$ of the aK-plane, where β and κ are two real parameters, and let $\bar{\boldsymbol{\gamma}}^{\beta,\kappa}$ be a natural parameterization of this

curve. We seek to prove that the functions $R_{\bar{\gamma}}^{\beta,\kappa}(s) = R(\bar{\gamma}_1^{\beta,\kappa}(s),\ \bar{\gamma}_2^{\beta,\kappa}(s))$ admit a unique global minimum.

First, let us remark that, for any point (a, K) of the domain Ω_R, there exists a unique curve that fulfill the two following conditions: (1) the curve pass through the point and (2), at this point, the derivative of R in the direction of the curve is zero or, equivalently, the curve is perpendicular to the gradient of R. Indeed, this is possible because the curve is parameterized by two parameters. Let $\beta^\star(a, K)$ and $\kappa^\star(a, K)$ be the parameters of the curve that fulfills the conditions.

Also, for any parameterization $\boldsymbol{\gamma}^{\beta,\kappa}$ of the curve, we define $t_\gamma^{\beta,\kappa}$ as the function that gives the parameter t from the coordinate of a point of the aK-plane. We have that

$$t = t_\gamma^{\beta,\kappa}(a, K) \quad \Leftrightarrow \quad \begin{cases} a = \gamma_1^{\beta,\kappa}(t) \\ K = \gamma_2^{\beta,\kappa}(t). \end{cases}$$

We also say that $\boldsymbol{\gamma}^{\beta,\kappa}$ is equivalent to $\bar{\boldsymbol{\gamma}}^{\beta,\kappa}$ if there exists a continuously differentiable bijective map ϕ, with $\phi'(t) \neq 0$ such that:

$$\bar{\boldsymbol{\gamma}}^{\beta,\kappa}(\phi(t)) = \boldsymbol{\gamma}^{\beta,\kappa}(t).$$

Lemma 7. *$R_{\bar{\gamma}}^{\beta,\kappa}$ admits a unique minimum for any real parameters β and κ if there exists a parameterization $\boldsymbol{\gamma}^{\beta,\kappa}$ equivalent to $\bar{\boldsymbol{\gamma}}^{\beta,\kappa}$ for which:*

$$\forall (a, K) \in \Omega_R,\ \beta = \beta^\star(a, K),\ \kappa = \kappa^\star(a, K),\ t = t_\gamma^{\beta,\kappa}(a, K),\ D_t^2 R_\gamma^{\beta,\kappa}(t) > 0.$$

Proof. Let us suppose there exists such a parameterization and let us prove that having a curve that admits more than one minimum is absurd. Since, we take $\beta = \beta^\star(a, K)$ and $\kappa = \kappa^\star(a, K)$, we have that the first derivative $D_t R_\gamma^{\beta,\kappa}(t)$ is zero and thus $R_\gamma^{\beta,\kappa}$ has an extremum in t. Since the second derivative is positive, then it is a minimum. Now assume that there exists another point (a', K') of the curve for the two conditions, *i.e.* such that $\beta^\star(a', K') = \beta$ and $\kappa^\star(a', K') = \kappa$. Then, R restricted to this curve would have at least two minima. But by continuity of the function, there should be a maximum between the two minimum which is absurd since there are no point at which the second derivative is negative. □

We do not provide any analytical proof. However, the second derivative that should be positive in the Lemma 7 can be computed numerically. We have thus verified that the lemma works for the parameterization $\boldsymbol{\gamma}^{\beta,\kappa}(t) =$

$(t, \beta t^3 + \kappa t)$, which is equivalent to the natural parameterization. Note that we have:

$$\begin{aligned} \beta^\star(a,K) &= -\frac{1}{2}\left[\frac{1}{a^2}\frac{\partial_a R}{\partial_K R} + \frac{K}{a^3}\right] \\ \kappa^\star(a,K) &= \frac{1}{2}\left[\frac{\partial_a R}{\partial_K R} + 3\frac{K}{a}\right] \end{aligned}$$

and

$$D_t := \frac{\partial a}{\partial t}\frac{\partial}{\partial a} + \frac{\partial K}{\partial t}\frac{\partial}{\partial K}$$

$$D_t^2 := \left(\frac{\partial a}{\partial t}\right)^2 \frac{\partial^2}{\partial a^2} + 2\,\frac{\partial a}{\partial t}\frac{\partial K}{\partial t}\frac{\partial^2}{\partial a\,\partial K} + \left(\frac{\partial K}{\partial t}\right)^2 \frac{\partial^2}{\partial K^2} + \frac{\partial^2 a}{\partial t^2}\frac{\partial}{\partial a} + \frac{\partial^2 K}{\partial t^2}\frac{\partial}{\partial K}.$$

Acknowledgement

This research project was funded in part by the Walloon Region under WIST 3 grant 1017074 (DOMHEX).

References

[1] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, et al. High-order CFD methods: current status and perspective. *International Journal for Numerical Methods in Fluids*, 72(8):811–845, 2013.

[2] R. M. Kirby, S. J. Sherwin, and B. Cockburn. To CG or to HDG: a comparative study. *Journal of Scientific Computing*, 51(1):183–212, 2012.

[3] P. E. J. Vos, S. J. Sherwin, and R. M. Kirby. From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low-and high-order discretisations. *Journal of Computational Physics*, 229(13):5161–5181, 2010.

[4] G. Karniadakis and S. Sherwin. *Spectral/hp element methods for computational fluid dynamics*. Oxford University Press, 2013.

[5] I. Babuška, B. A. Szabo, and I. N. Katz. The p-version of the finite element method. *SIAM Journal on Numerical Analysis*, 18(3):515–545, 1981.

[6] I. Babuška and B. Q. Guo. The h-p version of the finite element method for domains with curved boundaries. *SIAM Journal on Numerical Analysis*, 25(4):837–861, 1988.

[7] R. H. MacNeal. *Finite Elements*. CRC Press, 1993.

[8] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26, 2013.

[9] P. M. Knupp. On the invertibility of the isoparametric map. *Computer Methods in Applied Mechanics and Engineering*, 78(3):313–329, 1990.

[10] P. Knupp. Label-invariant mesh quality metrics. In *Proceedings of the 18th International Meshing Roundtable*, pages 139–155. Springer, 2009.

[11] P. L. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 90(9):1156–1182, 2012.

[12] A. Johnen, J.-F. Remacle, and C. Geuzaine. Geometrical validity of curvilinear finite elements. *Journal of Computational Physics*, 233:359–372, 2013.

[13] A. Johnen and C. Geuzaine. Geometrical validity of curvilinear pyramidal finite elements. *Journal of Computational Physics*, 299:124–129, 2015.

[14] D. A. Field. Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, 47(4):887–906, 2000.

[15] J. R. Shewchuk. What is a good linear finite element? interpolation, conditioning, anisotropy, and quality measures (preprint). Preprint, 2002.

[16] P. M. Knupp. Algebraic mesh quality metrics. *SIAM journal on scientific computing*, 23(1):193–218, 2001.

[17] P. M. Knupp. Algebraic mesh quality metrics for unstructured initial meshes. *Finite Elements in Analysis and Design*, 39(3):217–241, 2003.

[18] X. Roca, A. Gargallo-Peiró, and J. Sarrate. Defining quality measures for high-order planar triangles and curved mesh generation. In *Proceedings of the 20th International Meshing Roundtable*, pages 365–383. Springer, 2012.

[19] A. Gargallo-Peiró, X. Roca, J. Peraire, and J. Sarrate. Distortion and quality measures for validating and generating high-order tetrahedral meshes. *Engineering with Computers*, pages 1–15, 2014.

[20] A. Gargallo Peiró. *Validation and generation of curved meshes for high-order unstructured methods*. PhD thesis, Universitat Politècnica de Catalunya, 2014.

[21] C. D. Meyer. *Matrix analysis and applied linear algebra*. Siam, 2000.

[22] O. K. Smith. Eigenvalues of a symmetric 3×3 matrix. *Communications of the ACM*, 4(4):168, 1961.

[23] J. Schöberl. Netgen An advancing front 2d/3d-mesh generator based on abstract rules. *Computing and visualization in science*, 1(1):41–52, 1997.